UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Felipe Born de Jesus

AJUSTAMENTO DE PESOS PARA RATINGS DE MÚLTIPLOS CRITÉRIOS EM RECOMENDAÇÃO DE ITENS

Florianópolis

2017

Felipe Born de Jesus

AJUSTAMENTO DE PESOS PARA RATINGS DE MÚLTIPLOS CRITÉRIOS EM RECOMENDAÇÃO DE ITENS

Dissertação submetida ao Programa de Pós Graduação em Ciências da Computação para a obtenção do Grau de Mestre em Ciências da Computação.
Orientadora: Profa. Dra. Carina Friedrich Dorneles

Florianópolis

2017

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Jesus, Felipe Born de
Ajustamento de pesos para ratings de múltiplos critérios em recomendação de itens / Felipe Born de Jesus ; orientadora, Carina Friedrich Dorneles - SC, 2017.
78 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós Graduação em Ciência da Computação, Florianópolis, 2017.

Inclui referências.

1. Ciência da Computação. 2. Ajuste de pesos. 3. Esparsidade de dados. 4. Sistemas de recomendação. 5. Algoritmo genético. I. Dorneles, Carina Friedrich. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Felipe Born de Jesus

Ajustamento de pesos para ratings de múltiplos critérios em recomendação de itens

Esta dissertação foi julgada adequada para obtenção do título de mestre e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Florianópolis, 23 de fevereiro de 2017.

_____
Profª. Carina Friedrich Dorneles, Drª.
Coordenadora do Programa

Banca Examinadora:

_____
Profª. Carina Friedrich Dorneles, Drª.
Universidade Federal de Santa Catarina
Orientadora

_____
Prof. Leandro Krug Wives, Dr.
Universidade Federal do Rio Grande do Sul
(Videoconferência)

_____
Prof. Ronaldo dos Santos Mello, Dr.
Universidade Federal de Santa Catarina

_____
Prof. Roberto Willrich, Dr.
Universidade Federal de Santa Catarina

Dedico este trabalho principalmente à minha noiva. Dedico, também, aos meus sonhos e ao meu futuro.

AGRADECIMENTOS

Agradeço primeiramente à instituição que me permitiu chegar até aqui: a Universidade Federal de Santa Catarina. Também, ao apoio financeiro cedido pela Capes nos dois primeiros anos de mestrado. Um agradecimento especial ao Programação de Pós Graduação em Ciências da Computação, e aos servidores e bolsistas que trabalham no programa, pela excelência no trabalho.

Agradeço, principalmente, à minha noiva, Karine Speck, que está ao meu lado nos piores momentos, transformando-os em bons com um simples sorriso, mas ao mesmo tempo faz dos meus dias os melhores a serem vividos. Agradeço pela ajuda, pela força, pela inspiração e pela motivação. E, claro, não posso deixar de agradecer à gatinha que eu e a Karine adotamos, a Yunna, por me morder nos momentos mais inoportunos. Agradeço também aos meus sogros, Edson Luiz Speck e Rosi Meri Speck, por me acolherem sempre que preciso.

Também agradeço ao meu amigo João Francisco (Kot), que apesar da distância, está sempre me apoiando. Aos meus colegas e amigos do Lisa: Geomar, Juarez, Felipe Pinto, Filipe Silva, Cleto, além, é claro, do meu amigo Luan, que me prestou auxílio durante boa parte do meu trabalho. Aos meus colegas e amigos do CEPED e do LabTrans.

Aos professores da UFSC, em especial aos professores que fazem parte do Programa de Pós Graduação em Ciências da Computação, principalmente à professora Carina, que me orientou até aqui desde a graduação.

Aos meus pais, Adriano de Jesus e Magda Ballejo Born, pelo auxílio e incentivo para continuar seguindo em frente. Ao meu irmão, Marcelo Born de Jesus, pela amizade e companheirismo e por me ouvir quando preciso.

"An equation for me has no meaning unless it expresses a thought of God."

Srinivasa Ramanujan

RESUMO

A recomendação de itens para usuários é uma tarefa comum no cotidiano. Tal recomendação existe desde um vendedor de uma loja, quando sugere roupas ao cliente, até aplicativos como o Netflix, que sugere filmes para os usuários. Em sistemas de recomendação, é necessário cada vez mais levar em consideração diversos fatores para que sejam recomendados itens aos usuários. No contexto deste trabalho, tais fatores são os critérios, sendo que diversos deles devem ser levados em consideração na recomendação. Além disso, alguns desses critérios podem ser mais importantes que os demais e essa importância pode variar de usuário para usuário. Na literatura, é comum encontrar abordagens que, considerando avaliações prévias de usuários para determinados itens e os critérios desses itens, infere quais avaliações o usuário forneceria para outros itens e critérios de itens. Contudo, sabe-se que as bases de dados de avaliações de um sistema de recomendação costumam ser esparsas, ou seja, apenas uma pequena parcela das avaliações são conhecidas. Esse problema, conhecido como esparsidade de dados, faz com que seja lenta a predição de todas as avaliações possíveis. Neste trabalho, é proposto que se trabalhe com dois conceitos distintos de avaliações de itens: *ratings* explícitos, fornecidos explicitamente pelos usuários, e *ratings* implícitos, inferidos pelo sistema. Cada *rating* está relacionado a um critério e, para que seja possível identificar quais critérios são mais importantes, cada um dos critérios é relacionado a um peso. Para se ajustar os pesos dos critérios, é proposto um problema de otimização que é solucionado via algoritmo genético. Ao se utilizar ajuste de pesos, combinado com o uso de *ratings* implícitos e explícitos, é possível reduzir consideravelmente o tempo de ajuste do sistema de recomendação, bem como aumentar a precisão das recomendações em cenários mais esparsos. Em cenários menos esparsos, ainda é possível reduzir consideravelmente o tempo de ajuste do sistema, mantendo-se a precisão das recomendações com valores próximos à precisão de outros trabalhos da literatura.

Palavras-chave: Problema da esparsidade de dados. *Rating* implícito. *Rating* explícito. Ajuste de pesos. Algoritmo genético. Múltiplos critérios.

ABSTRACT

From a seller to applications like Netflix we can witness a recommendation. So, everyday there is examples of recommendation for a user. However, for a precise recommendation, a set of features are needed to be taken into account. Features like geographical localization, price of items, who acquired those items and so on. So, for a user, multiple criteria must be used in a recommendation. Moreover, some of those criteria may be more important than others. In order to identify which criteria are more important the others, we propose that each criterion is related to a weight. We propose, to identify those wheights, to adjust the weight of the criterion, by proposing an optimization problem that is solved by a genetic algorithm. Another problem we propose to mitigate is the data sparsity problem. In ratings database, usually there are only a few ratings, comparing to the total of possible ratings. In literature, an approach to mitigate this problem is, considering previous evaluations for items and items criteria from users, we can estimate which evaluation the users would provide to the items or the items criteria. Once there are a lot of ratings to estimate, the data sparsity problem makes the prediction for all possible evaluations to be slow. So, we propose the use of two different domains of ratings: explicit rating, explicitly provided by the user, and implicit rating, inferred by the system. Each rating is related with a criterion. Combining the weight adjustment with the use of implicit and explicit rating, it is possible to decrease the time needed for the recommender system adjustment, as well as increase the precision of recommendation in a sparse scenario. In a sparser scenario, when we remove evaluations maded by the users, it is still possible to decrease the time of adjustment, keeping the precision of our approach close to other baselines approaches.

Keywords: Data sparsity problem. Implicit rating. Explicit rating. Weighting adjustment. Genetic algorithm. Multi-criteria.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Em um sistema de recomendação, há duas maneiras de se obter as preferências de um usuário: através de *feedback* explícito ou através de *feedback* implícito (RICCI et al., 2010). No primeiro caso, o usuário explicita a avaliação de um item ou de um atributo desse item. No segundo caso, tal avaliação é inferida pelo próprio sistema de recomendação. Quando um usuário gosta de música e encontra um filme com uma ótima trilha sonora, por exemplo, tal usuário pode prover uma boa avaliação do filme, através de um *rating* explícito, resultando em um *feedback* explícito. Por outro lado, o sistema de recomendação pode, proativamente, com base em outras avaliações, identificar que um filme possui uma trilha sonora compatível com o usuário e então avaliar de forma positiva, através de um *rating* implícito, este filme a algum usuário que gosta de música. Essa avaliação resulta em *feedback* implícito.

Obtidas as preferências, um ponto que pode ser considerado na modelagem delas é o esquema de pesos (LAKIOTAKI K.; MATSATSINIS, 2011), em que cada usuário possui um conjunto de pesos, cada peso relacionado a determinado aspecto de um item (gênero de um filme, qualidade visual do filme, etc.). Um aspecto pode ser entendido como um critério (NILASHI; IBRAHIM; ITHNIN, 2014), atributo mensurável de um item que varia de acordo com cada usuário. A mensuração de um critério resulta em um *rating*. Adaptando-se o esquema de pesos em domínios de aplicação que utilizam critérios, é possível estabelecer o quão importante é cada critério para um usuário, uma vez que cada peso está associado a um critério. Todavia, em sistemas de recomendação, a porcentagem de *ratings* desconhecidos é muito alta, gerando o problema da **esparsidade de dados** (LIU; MEHANDJIEV; XU, 2011; ADOMAVICIUS; MANOUSELIS; KWON, 2011). Um exemplo real desta problemática pode ser encontrado na base de dados do MovieLens[1], onde existe um total de 22 milhões de *ratings* aplicados por 240 mil usuários para 33 mil filmes. A quantidade de pares usuário e filmes é de 7,92 bilhões (240 mil usuários para 33 mil filmes), sendo apenas 0,3% de *ratings* conhecidos, ou seja, mesmo uma base de dados com muitos *ratings* possui uma taxa pequena de avaliações conhecidas.

Para se entender o problema da esparsidade, basta considerar que dificilmente um usuário conhece todos os itens de determinada base de dados, tampouco dedica tempo fornecendo informações ou ava-

[1] `http://grouplens.org/datasets/movielens/`

liações aos itens ou aos critérios de um dado item. Isso faz com que a quantidade de ***ratings* explícitos** existentes no sistema de recomendação seja muito pequeno com relação à quantidade de *ratings* possíveis. Ainda assim, existem sistemas de recomendação que, mesmo utilizando poucos *ratings*, fazem recomendações precisas aos usuários. Para isso, são aplicadas técnicas que procuram solucionar o problema de esparsidade utilizando ***ratings* implícitos** (HUANG; CHEN; ZENG, 2004; MOSHFEGHI; PIWOWARSKI; JOSE, 2011; GUO, 2013; ZHANG et al., 2014b).

Em muitas dessas técnicas, como a filtragem colaborativa (NILASHI; IBRAHIM; ITHNIN, 2014; HWANG, 2010; JANNACH; KARAKAYA; GEDIKLI, 2012), quando um *rating* é desconhecido, ele é predito pelo sistema, com base em informações de *feedback* explícito fornecidas por outros usuários (PALANIVEL; SIVAKUMAR, 2010). No entanto, a abordagem de se predizer *ratings* usando abordagens colaborativas demanda tempo excessivo de processamento, o que pode trazer dificuldades em se aplicar tais técnicas em sistemas reais altamente esparsos. O tempo de processamento elevado se dá pela complexidade computacional linear (SARWAR et al., 2001) de se encontrar a similaridade entre pares de usuários, para só então se estimar os *ratings* desconhecidos de itens. Nota-se que técnicas de fatorização de matrizes (KOREN; BELL; VOLINSKY, 2009) podem ser utilizadas para otimizar o desempenho da similaridade entre usuários. Ainda assim, para se fazer a recomendação, é necessário, nas técnicas te filtragem colaborativa, predizer os *ratings* desconhecidos, o que demanda muito tempo computacional.

Portanto, no presente trabalho propõe-se o uso dos dois conceitos de *ratings*, com a finalidade de se reduzir a quantidade de processamento necessário em uma recomendação: explícitos, fornecidos explicitamente pelos usuários para itens (ou critérios de itens); e implícitos, computados para pares de usuário e itens (ou critérios de itens). É importante observar que *ratings* explícitos podem ser desconhecidos, pois nem sempre usuários fornecem avaliações aos itens, mas *ratings* implícitos sempre serão conhecidos, uma vez que podem ser computados pelo sistema para quaisquer pares usuário-item. Portanto, a abordagem proposta neste trabalho utiliza uma combinação de ***ratings* explícitos**, quando existentes, com ***ratings* implícitos**, sempre existentes. Reduz-se a quantidade de *ratings* preditos pelo fato de apenas serem computados *ratings* implícitos, não se predizendo *ratings* explícitos desconhecidos, ou seja, diferente do proposto por trabalhos relacionados, apenas *ratings* implícitos são calculados.

Outro ponto importante na recomendação de itens é saber qual a

importância de cada critério para o usuário. Isso possibilita que sejam fornecidas recomendações personalizadas aos usuários. Por exemplo, pode ser que uma recomendação seja mais precisa para determinado usuário se ela considerar que o critério mais importante é o de `efeitos visuais`. Para que seja possível identificar diferentes pesos para diferentes critérios, no presente trabalho, é usada uma abordagem em que cada usuário possui um conjunto de pesos, estando cada peso relacionado a um critério. A geração desse conjunto de pesos se dá, inicialmente, com pesos aleatórios e, a partir daí, eles são ajustados de acordo com os *ratings* já conhecidos fornecidos para o usuário, utilizando-se um algoritmo genético. A ideia é que *ratings* altos para critério podem indicar sua importância ao usuário. Na literatura, alguns trabalhos que propõem o ajuste de pesos, usando regressão múltipla (MONTGOMERY, 2008) e técnicas baseadas em gradiente descendente (RENDLE et al., 2009), também foram propostos. No entanto, no caso da regressão múltipla, o problema é o fato de serem gerados pesos negativos, enquanto técnicas como gradiente descendente possuem tempo muito variável para se obter otimizações. O algoritmo genético, entretanto, possibilita que sejam adicionadas restrições aos pesos e aos conjuntos de pesos, por exemplo, ao mesmo tempo que permite ser definida uma quantidade máxima de iterações.

Há dois problemas, portanto, a serem tratados neste trabalho: (i) reduzir os efeitos da esparsidade, mesmo existindo uma alta taxa de *ratings* desconhecidos, obtendo-se precisão comparável às demais técnicas da literatura sem a necessidade de se predizer tais *ratings*; e (ii) gerar adequadamente os pesos dos critérios para cada usuário. Para o primeiro problema, é proposta uma abordagem em que se combina *ratings* explícitos com *ratings* implícitos. Para o segundo problema, é proposta uma abordagem que utiliza algoritmo genético.

As principais contribuições deste trabalho são as seguintes: (i) geração de *ratings* implícitos para a mitigação do problema de esparsidade de dados, evitando a necessidade do uso de filtragem colaborativa para predizer *ratings* explícitos desconhecidos; (ii) algoritmo genético que ajusta pesos, estimando quão importante é cada critério para determinado usuário; uma função de *fitness* é definida, a fim de representar quão bom é um dado conjunto de pesos a um determinado usuário. A noção de "bom" é baseada em quão próximo um *rating* predito é de um *rating real*; e (iii) realização de experimentos em uma base de dados pública e real, a MovieLens (seguindo a linha de importantes trabalhos da literatura, tal como (ZHANG et al., 2014a)), através da comparação com quatro *baselines*. Os experimentos mostram bom desempenho

em termos de precisão e uma economia relevante de tempo gasto para predição de *ratings*.

Este trabalho está organizado da seguinte maneira. Primeiramente, introduz-se um exemplo motivacional e são apresentados os principais conceitos para o entendimento do problema e da proposta no Capítulo 2. No Capítulo 3 são discutidos os trabalhos relacionados. No Capítulo 4, a proposta é apresentada, tratando os problemas de esparsidade de dados e de geração dos pesos dos critérios para cada usuário. No Capítulo 5 são apresentados os experimentos executados para avaliação da proposta, com ênfase nas contribuições deste trabalho. Conclusões e trabalhos futuros são discutidos no Capítulo 6.

2 EXEMPLO MOTIVACIONAL E CONCEITOS BÁSICOS

Neste capítulo são mostrados os fundamentos da proposta deste trabalho, ao se introduzir um exemplo motivacional na seção 2.1; e então serem definidos os conceitos que são utilizados ao longo deste trabalho, no subcapítulo 2.2.

2.1 EXEMPLO MOTIVACIONAL

Nesta seção é discutido um cenário motivacional de recomendação de filmes que ilustra tanto o uso do conjunto de pesos quanto a predição de *ratings* implícitos. A Figura 1 apresenta um exemplo de avaliação de um filme do usuário Bob, com *ratings* implícitos e explícitos. Nesse exemplo, os *ratings* explícitos fornecidos, representados por avaliações entre 1 e 5 estrelas, estão associados aos critérios `Atuação`, `Qualidade Visual`, `Qualidade sonora` e `Enredo`. Os *ratings* implícitos, que são números no intervalo entre 0 e 1, computados pelo sistema, estão associados com os critérios `Data de lançamento` e `Gênero`.

Como é comum em um sistema de recomendações, Bob não forneceu *ratings* (para `Qualidade sonora`, por exemplo) para todos os critérios de *Star Wars*. Bob não forneceu, portanto, *ratings* para todos os filmes do sistema. Como exemplificado na Tabela 1, para os Filmes 3 e 4, Bob não forneceu nenhuma avaliação explícita, gerando uma matriz esparsa. Por outro lado, considerando apenas os *ratings* implícitos, é possível a obtenção de uma matriz completa. Vale destacar que *ratings* com valor 0 são *ratings* conhecidos, mas que representam que o item não possui relevância alguma para o usuário. Assim, quando o sistema precisar utilizar os *ratings* de um item para uma recomendação, no mínimo existirão aqueles *ratings* computados implicitamente pelo sistema, normalmente gerados através da análise do comportamento do Bob, ou seja, pelo seu *feedback* implícito. Nota-se, portanto, que os *ratings* implícitos são uma maneira de mitigar o problema de esparsidade de dados, servindo como base para solução de um dos problemas destacados neste trabalho.

Além dos *ratings* existentes para o usuário Bob, é importante considerar quais critérios são mais importantes para ele, de forma a se gerar uma recomendação personalizada. Pela Tabela 1, considerando cada uma das colunas como um critério, tem-se que o conjunto de pesos

Tabela 1 – Matriz de *ratings*

Bob	Ratings Explícitos				Ratings Implícitos		Rating do Item
	Acting	Visual Quality	Audio Quality	Story	Genre	Release Date	
Star Wars	0.4	0.8	?	0.6	0.8	0.9	0.8
Filme 2	0.5	0.8	0	0.7	0	0.7	0.9
Filme 3	?	?	?	?	0	0.9	?
Filme 4	?	?	?	?	0.5	0	?
Pesos	0.15	0.20	0.05	0.10	0.30	0.20	N/A

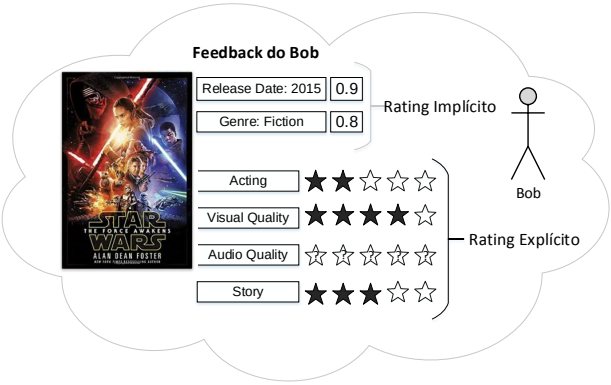

Figura 1 – *Ratings* implícitos e explícitos em um item com múltiplos critérios

do Bob é {0,15; 0,20; 0,05; 0,10; 0,30; 0,20}. É possível inferir, a partir desse conjunto, que o critério `Genre` é o de maior interesse para o Bob, enquanto `Audio Quality` é o de menor peso. Como cada peso está relacionado a um critério, existem pesos associados a *ratings* desconhecidos, como é o caso de `Qualidade sonora` no filme `Star Wars`. Dessa forma, o problema é: "o que fazer quando há *ratings* desconhecidos para o Bob?". Mais especificamente, como calcular um *rating* global para o filme `Star Wars`, por exemplo, já que `Audio Quality` é desconhecido? $(0,4 \cdot 0,15 + 0,8 \cdot 0,20 + \mathbf{?} \cdot 0,05 + 0,6 \cdot 0,10 + 0,8 \cdot 0,30 + 0,9 \cdot 0,20) \approx \mathbf{?}$. A solução adotada é ignorar o peso relacionado aos *ratings* explícitos desconhecidos e reajustar os demais, mantendo-se a mesma proporção. Por exemplo, no caso da recomendação de `Star Wars`, como o *rating* desconhecido está relacionado a um critério de peso $0,05$, é necessário ajustar os demais pesos dividindo-os por $0,95$: $\frac{(0,4 \cdot 0,15 + 0,8 \cdot 0,20 + 0,6 \cdot 0,10 + 0,8 \cdot 0,30 + 0,9 \cdot 0,20)}{0,95} \approx 0,74$. Para os demais filmes da tabela, ter-se-iam os seguintes *ratings* finais dos items: `Filme 2`: $(0,5 \cdot 0,15 + 0,8 \cdot 0,20 + 0,0 \cdot 0,05 + 0,7 \cdot 0,10 + 0,0 \cdot 0,30 + 0,7 \cdot 0,20) \approx \mathbf{0,445}$; `Filme 3`: $\frac{(0,0 \cdot 0,30 + 0,9 \cdot 0,20)}{0,5} \approx \mathbf{0,36}$; e `Filme 4`: $\frac{(0,5 \cdot 0,30 + 0,0 \cdot 0,20)}{0,5} \approx \mathbf{0,30}$. Com esses itens e *ratings*, a ordem de recomendação para o Bob seria a seguinte: `Star Wars` (0,74), `Filme 2` (0,445), `Filme 3` (0,36) e `Filme 4` (0,30).

2.2 CONCEITOS BÁSICOS

Antes de dar início à descrição da proposta, é importante que alguns conceitos sejam definidos a fim de deixar a compreensão livre de ambiguidades. Os principais conceitos definidos são: item, múltiplos critérios, *ratings* explícitos e implícitos, *feedback* implícito e explícito, e pesos de critérios.

Definição 1 ***Item e múltiplos critérios:*** *Seja* $I = \{i_1, \cdots, i_j\}$ *o conjunto de* j *itens e* $C = \{c_1, \cdots, c_k\}$ *o conjunto de* k *critérios, onde cada critério é um rótulo representando um aspecto avaliável de um item. Cada item* $i \in I$ *possui um conjunto de critérios* $C_i = \{c_{i1}, \cdots, c_{il}\}$.

A Figura 1 apresenta um exemplo de itens e seus critérios. O item $i =$ `Star Wars` possui o conjunto de 6 critérios $C_{StarWars} =$ {`Acting, Visual Quality, Audio Quality, Story, Genre, Release Date`}.

Definição 2 ***Rating explícito e rating implícito:*** *Rating explícito* $r^{exp} \in \{\{\emptyset\} \cup \{\mathbb{R} | 0 \leq r^{exp} \leq 1\}\}$ *é o rating provido explicitamente por um usuário. Rating implícito* $r^{imp} \in \{\mathbb{R} | 0 \leq r^{imp} \leq 1\}$ *é o rating computado implicitamente para algum usuário.*

Pela definição, a principal diferença entre um *rating* implícito e um *rating* explícito está no fato deste último ter a possibilidade de ser desconhecido, quando $r^{exp} = \emptyset$. Isso não acontece com o *rating* implícito, uma vez que ele pode ser calculado. É importante observar que r^{imp} pode ser 0, o que é diferente de um *rating* desconhecido $\emptyset$.

Definição 3 ***Feedback explícito e feedback implícito:*** *Feedback é uma tripla* (u, r, λ)*, na qual* u *é um usuário,* r *é um rating e* λ *pode ser tanto um item* i *quanto um critério* c_i *do item* i*. O conjunto* Fb^{exp} *é o conjunto de feedback explícito, quando* $r = r^{exp}$*, enquanto* Fb^{imp} *é o conjunto de feedback implícito, quando* $r = r^{imp}$.

Através da Definição 3, é possível notar que podem haver *feedbacks* explícitos, cujos *ratings* são explicitamente providos por um usuário, ou *feedbacks* implícios, cujos *ratings* são calculados pelo sistema. Outra característica importante apresentada na Definição 3 é que uma tripla de *feedback* pode estar relacionada tanto a um item quanto a um critério de um item. Observando a Figura 1, é possível

observar que a tupla $Fb^{imp} = \{Bob, 0.8, Genre_{StarWars}\}$ refere-se ao critério `Genre` do item `Star Wars`. Já observando a Tabela 1, é possível observar que a tupla $Fb^{imp} = \{Bob, ?, Filme4\}$ refere-se ao item `Filme 4`, sem *rating* fornecido.

Definição 4 ***Pesos dos critérios****: Seja $W_u = \{\omega_1, \cdots, \omega_k\}$ o conjunto de pesos para o usuário u, onde k é o número de critérios e cada ω_j é um par (w, c), em que $w \in \mathbb{R}$ e $w \geq 0$ é um peso e $c \in C$ é um critério, tal que $\sum_{\omega_j=(w,c)\in W_u}(w) = 1$.*

A Definição 4 impõe o conjunto de pesos para um determinado usuário, considerando os critérios de um item. Assim, considerando o exemplo da Tabela 1, onde estão os exemplos de pesos para cada critério do item `Star Wars`, o usuário Bob teria o conjunto W_{Bob} ={(0.15, Acting); (0.20, Visual Quality); (0.05, Audio Quality); (0.10, Story); (0.30, Genre); (0.20, Release Date)}, sendo que somatório é igual a 1.

3 TRABALHOS RELACIONADOS

Personalização de ranqueamento e abordagens utilizando múltiplos critérios não são novidades nas literaturas de sistemas de recomendação e recuperação de informação. Todavia, ainda existem características inexploradas quando se considera o problema de *rating* de múltiplos critérios, como a possibilidade de serem combinados *ratings* implícito e *ratings* explícitos, reduzindo o efeito negativo que a esparsidade de dados tem sobre as recomendações.

Existem trabalhos que utilizam a abordagem de múltiplos critérios para melhorar a precisão de resultados (WANG; KWONG, 2014; SZELAG; GRECO; SŁOWIŃSKI, 2014; NILASHI; IBRAHIM; ITHNIN, 2014), além de, pelo menos dois, *surveys* publicados na literatura que mencionam os problemas de múltiplos critérios. O primeiro deles (MANOUSELIS; COSTOPOULOU, 2007) descreve diferentes categorias de abordagens de múltiplos critérios, enquanto o segundo (ADOMAVICIUS; MANOUSELIS; KWON, 2011) estuda diferentes abordagens de múltiplos critérios, em especial o problema de *ratings* de múltiplos critérios, ressaltando tarefas inexploradas na literatura de sistemas de recomendação. Aplicações contemporâneas também consideram múltiplos critérios no processo de ranqueamento para recomendações, como no Netflix[1] por exemplo, que considera os atores dos filmes ou seriados assistidos pelos usuários, bem como os gêneros desses filmes ou seriados e quanto tempo foi passado assistindo-lhes.

Além de se utilizar múltiplos critérios, existem trabalhos que combinam técnicas para fornecer recomendações, como é o caso de Lamontagne et al.(LAMONTAGNE; ABI-ZEID, 2006), ao combinar múltiplas métricas de similaridade para sistemas de recomendação, sem precisar considerar nenhum critério. Além desse trabalho citado, outros combinam mútiplas técnicas em outros escopos, como Lee (LEE; CROFT, 2014), que combina múltiplas técnicas para recuperar informações textuais de diferentes linguagens, ou o trabalho de Rousseau (ROUSSEAU; VAZIRGIANNIS, 2013), que ajusta pesos para grafos considerando diferentes parâmetros de TF-IDF.

A abordagem do presente trabalho, ao distinguir *ratings* explícitos de implícitos, possibilita que *ratings* explícitos não precisem ser preditos. Tal abordagem se diferencia de abordagens que predizem *ratings* através de filtragem colaborativa (NILASHI; IBRAHIM; ITHNIN, 2014; HWANG, 2010; JANNACH; KARAKAYA; GEDIKLI, 2012). Uma das

[1] https://www.netflix.com

vantagens de não se utilizar a filtragem colaborativa é economia de tempo de processamento nas predições. O ajuste de pesos proposto no presente trabalho para os critérios utiliza um algoritmo genético, assim como (HWANG, 2010), com a diferença de que o algoritmo proposto por (HWANG, 2010) é projetado para um número fixo de 4 pesos, enquanto o presente trabalho considera um conjunto de pesos com número ilimitado de critérios. Outras abordagens para ajuste de pesos incluem otimização via gradiente descendente (RENDLE et al., 2009), que tem como desvantagem não obter valores ótimos globais de otimização, mas sim valores ótimos locais, segundo (HAYKIN; NETWORK, 2004) - o que é evitado em algoritmo genético, por causa da operação de mutação -, e a regressão múltipla (MONTGOMERY, 2008), que pode ter como resultado pesos não normalizados. Para se contornar tal problema, poder-se-ia imaginar uma abordagem de normalização dos pesos, adotando-se uma estratégia de proporção entre os valores. Contudo, o resultado da função de maximização, nessa estratégia, é diferente do resultado utilizando-se os valores absolutos. Como o algoritmo genético permite manter sempre a restrição dos pesos normalizados, em cada interação, tal problema não ocorre.

Nas seções seguintes são comentados trabalhos relacionados a tópicos importantes desta proposta, como múltiplos critérios, ajuste de pesos, esparsidade de dados e trabalhos com propostas de *ratings* implícitos, além de uma seção com tabela comparativa dos principais trabalhos.

3.1 MÚLTIPLOS CRITÉRIOS

Existem aplicações, como Netflix `https://www.netflix.com/`, que para ranquear recomendações consideram múltiplos critérios, como atores dos filmes que foram assistidos [2], bem como o gênero de tal filme e por quanto tempo cada filme foi visto. Para contemplar a necessidade de se considerar múltiplas técnicas para recomendar, alguns trabalhos foram feitos, como de (LAMONTAGNE; ABI-ZEID, 2006), cuja contribuição é a combinação de múltiplas métricas de similaridade, mas com restrição de que é extensível para apenas métricas de similaridade. Outros trabalhos que combinam múltiplas técnicas, como o trabalho de (LEE; CROFT, 2014), que combina diferentes abordagens de recuperação de informação para retornar informações em textos com conteúdo

[2] Informações retiradas da página do Netflix: `https://help.netflix.com/pt/node/9898`

em diferentes idiomas, ou em (ROUSSEAU; VAZIRGIANNIS, 2013), que utiliza múltiplas técnicas para ajustar peso a um grafo com diferentes parâmetros para o TF-IDF.

Em outros trabalhos, como (HWANG, 2010; JANNACH; KARAKAYA; GEDIKLI, 2012; NILASHI; IBRAHIM; ITHNIN, 2014; SZELAG; GRECO; SŁOWIŃSKI, 2014; WANG; KWONG, 2014), o foco é recomendação baseada em *ratings* de múltiplos critérios. As abordagens de (HWANG, 2010; JANNACH; KARAKAYA; GEDIKLI, 2012; NILASHI; IBRAHIM; ITHNIN, 2014) consideram *ratings* para cada critério, sendo tais *ratings* fornecidos explicitamente por usuários, de modo que *ratings* desconhecidos são computados. Na abordagem de (SZELAG; GRECO; SŁOWIŃSKI, 2014), os *ratings* considerados servem para decidir como ranquear os itens em uma abordagem conhecida como *Multiple Criteria Decision Maker(MCDM)*. Outro trabalho que faz uso de MCDM é do (WANG; KWONG, 2014), em que um *decision maker* computa parâmetros para uma técnica de *machine learning* aprender as características, considerando múltiplos critérios, de forma que essa *machine learning* se adapte a novas entradas. Tal trabalho utiliza múltiplos critérios, portanto, para as características que determinada *machine learning* precisa aprender.

3.2 AJUSTE DE PESOS

O ajuste de pesos está normalmente relacionado com o problema de otimização, em que dado um conjunto de características, deseja-se saber a importância de cada característica para uma função objetivo. No contexto do presente trabalho, as características são os critérios, enquanto a função objetivo é a função de maximização. Da mesma forma, a proposta de (HORN; NAFPLIOTIS; GOLDBERG, 1994; HWANG, 2010) faz uso de algoritmo genético para ajustar pesos.

Outras abordagens de ajuste de peso para múltiplos critérios incluem *Multiple Criteria Decision Maker*, como mostrado em (WANG; KWONG, 2014), em que o ranqueamento é feito a partir da computação da probabilidade de um item ser mais importante que outro, baseado em cada critério; redes neurais, como em (HAYKIN; NETWORK, 2004), que otimiza critérios baseado em funções simples, cujas derivadas sejam conhecidas e, desse modo, o cálculo matemático para obter valores ótimos é trivial. *Support vector machine (SVM)*, como proposto por (JOACHIMS, 1999, 2002), cujo objetivo é encontrar linhas ou planos que separam duas classes de dados, o que auxilia em propostas envolvendo

agrupamento. Também há a possibilidade de otimização de programação linear com múltiplos critérios, como no trabalho de (HORST et al., 2007).

3.3 ESPARSIDADE DE DADOS

A esparsidade de dados não é um problema recente e, na verdade, é algo até mesmo natural. Observando-se alguns exemplos cotidianos, é comum que uma pessoa vá a uma livraria, compre alguns livros, leia-os e converse com seus amigos a respeito. Seus amigos, por outro lado, podem ter lido alguns desses livros, possivelmente outro. O ponto é que certamente nenhum desses amigos leu todos os livros existentes. Ainda assim, para diversos trabalhos, como (NILASHI; IBRAHIM; ITHNIN, 2014; HWANG, 2010) e até mesmo para o presente trabalho, quanto menos esparsos forem os dados, melhores são os resultados de recomendação.

O trabalho de (ZHANG et al., 2014c) foca nos problemas relativos à esparsidade, citando outras aplicações que podem sofrer com esse efeito. Antes desse trabalho, contudo, outros trabalhos, como (HUANG; CHEN; ZENG, 2004; MOSHFEGHI; PIWOWARSKI; JOSE, 2011; GUO, 2013) tratam e mitigam o problema da esparsidade de dados em sistemas de recomendação, sendo os dois primeiros no contexto de filtragem colaborativa e o terceiro em sistemas de recomendação que sofrem com a partida lenta.

3.4 CÁLCULO DE *RATINGS* IMPLÍCITOS

ratings implícitos são os *ratings* que precisam ser computados, ou seja, não foram diretamente fornecidos pelo usuário. Existem alguns trabalhos que computam *ratings* e que podem ser usados, dependendo do contexto.

O trabalho de (RENDLE et al., 2009) computa *ratings* de acordo com alguns *feedbacks* implícitos do usuário, tais como cliques nas páginas e tempo de visualização de cada item na página. (PALANIVEL; SIVAKUMAR, 2010) oferece uma série de técnicas que usam informações de *feedback* implícito. Em aplicações que guardam tais informações é possível, portanto, utilizá-los. O trabalho de (MINKOV et al., 2010), por outro lado, recomenda eventos que tenham duração e ocorrência pré determinadas, como seminários ou conferências, o que deixa a recomendação limitada, visto que após um evento ocorrer, tal evento não

deve mais ser recomendado, visto que já aconteceu. Para isso, propõe a geração de *ratings* que levam em conta a duração dos eventos e outras características. Outro trabalho que utiliza características temporais é o de (COSTA; COUTO; SILVA, 2014). Em aplicações que possuem *tags* para os itens ou informações de relacionamento entre usuários é possível utilizar as técnicas de (HU et al., 2012; GEDIKLI; JANNACH, 2013). Outros trabalhos que fazem uso de informações de relacionamento são os de (MA, 2014; SARWAR et al., 2001).

3.5 TABELA COMPARATIVA

Para ressaltar as contribuições do presente trabalho sobre os trabalhos relacionados, é apresentada uma visão geral destes trabalhos na Tabela 2. A proposta do presente trabalho foca na geração de um ranqueamento de qualquer tipo de item, com suporte para múltiplos critérios, sendo que tais critérios possuem pesos personalizados. Além disso, é proposto um tratamento para o problema da esparsidade de dados, além de lidar com *ratings* implícitos e explícitos.

Na Tabela 2, as colunas indicam as características que os trabalhos possuem, enquanto as linhas representam os trabalhos propriamente ditos. As características ressaltadas são: tipo de item, múltiplos critérios, ajuste de pesos, esparsidade de dados e tipo de *rating*. Abaixo são descritas as informações referentes a cada coluna:

- **Tipo de item:** indica para qual tipo de item o trabalho foca. Por exemplo, se a abordagem proposta serve para qualquer item (é uma abordagem genérica) ou apenas para determinado tipo de item (como artigo científico ou filme). A abordagem descrita no presente trabalho serve para quaisquer tipos de itens.

- **Múltiplos critérios:** indica se o trabalho trata de múltiplos critérios e se existe alguma limitação com relação ao número máximo de critérios. A abordagem descrita no presente trabalho utiliza múltiplos critérios.

- **Ajuste de pesos:** indica qual a abordagem utilizada para ajuste de pesos, quando o trabalho trata de ajuste de pesos. A abordagem descrita no presente trabalho utiliza algoritmo genético para ajuste de pesos.

- **esparsidade de dados:** indica se a abordagem é afetada pelo problema de esparsidade de dados, se não é afetada ou se, apesar

de ser afetada, mitiga os efeitos. No contexto do presente trabalho, a esparsidade é tratada, mitigada, mas ainda assim afeta o desempenho (em termos de precisão) das recomendações.

- **Tipo de *rating*:** indica se a abordagem proposta dos trabalhos utiliza *rating* implícito, explícito ou ambos. O presente trabalho trata tanto de *rating* implícito quanto de *rating* explícito.

Os trabalhos comparados com o presente são Lamontagne'06 (LAMONTAGNE; ABI-ZEID, 2006), Lee'14 (LEE; CROFT, 2014), Nilashi'14 (NILASHI; IBRAHIM; ITHNIN, 2014), Costa'14 (COSTA; COUTO; SILVA, 2014), Minkov'10 (MINKOV et al., 2010), Hwang'10 (HWANG, 2010), Szelkag'14 (SZELAG; GRECO; SŁOWIŃSKI, 2014), Wang'14 (WANG; KWONG, 2014), Cheng'14 (CHENG et al., 2014), Rendle'09 (RENDLE et al., 2009), Kharitonov'13 (KHARITONOV et al., 2013) e Li'13 (LI; YANG; ZHANG, 2013). Os aspectos que diferenciam esta proposta dos demais trabalhos relacionados são os seguintes:

- **Lamontagne'06:** combina múltiplas métricas de similaridade para recomendação de itens. Supondo que cada métrica retorne um *rating* implícito, é possível se considerar que tal trabalho utiliza múltiplos critérios. Além disso, a abordagem desse trabalho serve para qualquer tipo de item. Contudo, a esparsidade de dados não afeta o desempenho em termos de recomendação, uma vez que são utilizadas métricas de similaridade que não dependem de *feedbacks* dos usuários. Todavia, esse trabalho não prevê ajuste de pesos nem mesmo *rating* explícito.

- **Lee'14:** embora contenha múltiplos critérios, utiliza-os para recuperação de documentos textuais que possam conter línguas distintas. Ele não prevê abordagem de ajuste de pesos. Os *ratings* gerados são implícitos e a esparsidade de dados não afeta o desempenho.

- **Nilashi'14:** utiliza um ajuste de pesos para múltiplos critérios com uma proposta baseada em técnica Neuro-Fuzzy. Os *ratings* dos critérios são todos explícitos e, pelo fato de o trabalho precisar prever *ratings* desconhecidos, é largamente afetado pela esparsidade de dados, não propondo até então nenhum tratamento para ele.

- **Costa'14:** utiliza múltiplos critérios, com ajuste de pesos baseado em em *Support Vector Machine (SVM)*. O SVM foi proposto

por (JOACHIMS, 1999, 2002). Os *ratings* gerados pela proposta são implícitos. Contudo, esse trabalho funciona apenas para itens com características temporais. Pelo fato de ser uma técnica específica para esse tipo de itens, o próprio trabalho já aborda técnicas para mitigação dos efeitos de esparsidade de dados.

- **Minkov'10:** propõe uma técnica que funciona apenas para eventos com duração limitada. Tal técnica, quando aplicada, gera um *rating* implícito. O efeito da esparsidade de dados é tratado, uma vez que o sistema necessita recomendar rapidamente os eventos para os usuários. Ainda assim, a técnica não aborda múltiplos critérios, tampouco ajuste de pesos.

- **Hwang'10:** propõe um algoritmo genético para ajuste de pesos de múltiplos critérios para recomendação de itens. A abordagem proposta tem como limitação 4 critérios, embora seja possível estender a técnica para mais critérios. Esse trabalho, contudo, necessita de *ratings* explícitos e, quando eles são desconhecidos, é necessário predizê-los, fazendo com que a esparsidade de dados afete o desempenho da proposta.

- **Szelkag'14:** utiliza um *Multi criteria decision maker (MCDM)* para recomendação de itens. A abordagem proposta serve para qualquer tipo de item, contudo os *ratings* dos critérios dos itens são explícitos e não há tratamento para o problema de esparsidade de dados.

- **Wang'14:** assim como o de (SZELAG; GRECO; SŁOWIŃSKI, 2014), utiliza um *MCDM* e é afetado pelo problema de esparsidade de dados. Não prevê também o uso de *ratings* implícitos em sua abordagem.

- **Cheng'14:** utiliza uma técnica de ranqueamento chamada *IMRank*. Tal ranqueamento é afetado pelo problema de esparsidade de dados e não considera múltiplos critérios. O *rating* gerado para ranqueamento é implícito.

- **Rendle'09:** propõe uma técnica que utiliza *feedbacks* implícitos dos usuários para recomendação de item. O uso desses *feedback* reduz os efeitos do problema de esparsidade de dados. Todavia, tal trabalho não prevê uso de múltiplos critérios e não adota nenhuma técnica para ajuste de pesos.

- **Kharitonov'13:** utiliza *feedbacks* advindos de cliques do usuário. Tal técnica não é afetada com o problema de esparsidade de dados. Contudo, a técnica proposta não adota múltiplos critérios nem ajuste de pesos.

- **Li'13:** propõe uma técnica de recomendação de artigos científicos baseado em múltiplas características. Tais características são mensuradas com *rating* implícito. Esse trabalho não é afetado com o problema de esparsidade de dados e, mas não prevê ajuste de pesos para os critérios e a técnica é utilizada para artigos científicos.

Tabela 2 – Tabela Comparativa

Trabalho	Tipo de item	Múltiplos Critérios	Ajuste de pesos	esparsidade de dados	Tipo de rating
Lamontagne'06	Genérico	Sim	Não	Não afetado	Implícito
Lee'14	Documentos textuais	Sim	Não	Não afetado	Implícito
Nilashi'14	Genérico	Sim	Neuro-Fuzzy	Afetado	Explícito
Costa'14	Itens com características temporais	Sim	SVM	Tratado	Implícito
Minkov'10	Eventos com duração limitada	Não	Não	Tratado	Implícito
Hwang'10	Genérico	Limitado (4)	Algoritmo Genético	Afetado	Explícito
Szelkag'14	Genérico	Sim	MCDM	Afetado	Explícito
Wang'14	Genérico	Sim	MCDM	Afetado	Explícito
Cheng'14	Genérico	Não	Não	Afetado	Implícito
Rendle'09	Genérico	Não	Não	Tratado	Implícito
Kharitonov'13	Genérico	Não	Não	Não afetado	Implícito
Li'13	Artigos Científicos	Sim	Não	Não afetado	Implícito
Esta proposta	Genérico	Sim	Algoritmo Genético	Tratado	Explícito e Implícito

4 AJUSTE DE PESOS PARA *RATINGS* DE MÚLTIPLOS CRITÉRIOS

Este capítulo apresenta a proposta do trabalho, primeiramente com uma visão geral da arquitetura na seção 4.1, onde são detalhados os componentes utilizados para implementação desta proposta. Em seguida, é mostrado como é tratado o problema de esparsidade de dados na seção 4.2. Finalmente, a seção 4.3 detalha a proposta para ajuste de pesos, incluindo uma proposta de algoritmo genético que resolve tal ajuste.

4.1 VISÃO GERAL DA ARQUITETURA

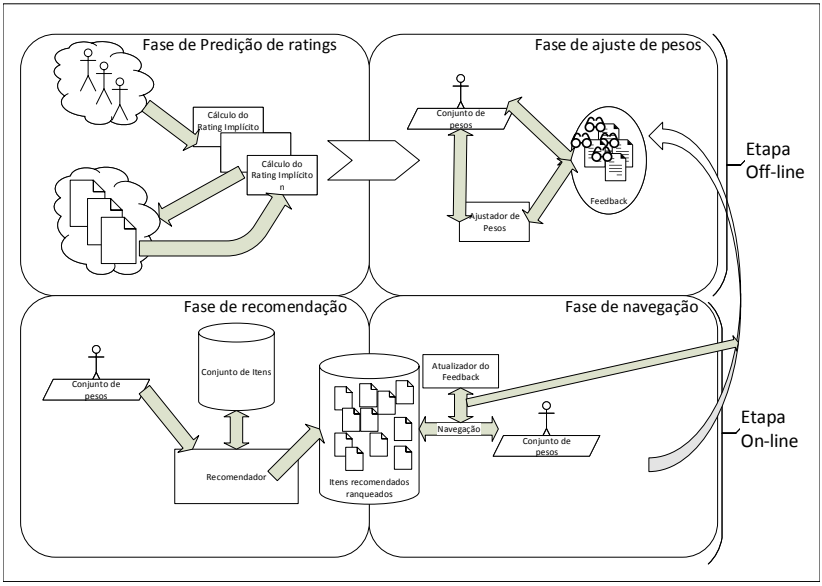

Figura 2 – Visão geral da arquitetura proposta

A Figura 2 apresenta uma arquitetura para a visão geral de um sistema de recomendação que utiliza as propostas deste trabalho. Tal arquitetura é composta por duas etapas principais: a etapa *offline* e a etapa *online*. A fase *offline* é responsável por deixar o sistema de

recomendação preparado para fazer as recomendações e subdivide-se em duas novas etapas: etapa de predição de *ratings* de relevância e fase de ajuste de pesos. A etapa *online* é a responsável pela recomendação e coleta de dados, subdividindo-se em fase de busca e fase de navegação. O detalhamento dessas sub-fases da arquitetura é o seguinte.

- **Fase de predição de *ratings* de relevância (*offline*)**: nesta fase as informações dos usuários são utilizadas para cálculo dos *ratings* implícitos.
- **Fase de ajuste de pesos (*offline*)**: com base nos *feedbacks* explícitos dos usuários e nos *ratings* calculados pela fase anterior, são computados os pesos dos critérios para cada usuário.
- **Fase de recomendação (*online*)**: o sistema mostra recomendações aos usuários. Utiliza-se, para recomendação, os pesos dos usuários (computados na fase anterior).
- **Fase de navegação (*online*)**: o usuário interage com o sistema. Nesta fase são capturadas as informações implícitas (como cliques, tempo de visualização, etc.) das navegações. É nesta fase também que o usuário fornece *feedbacks* explícitos a respeito das recomendações feitas pelo sistema.

Com base na arquitetura descrita, é possível identificar quatro componentes: *Recomendador*, *Ajustador de pesos*, *Estimador de ratings* e *Ordenador do ranqueamento*, como ilustrado na Figura 3, sendo os componentes *Recomendador* e *Ordenador do ranqueamento* pertencentes à etapa *online* e os componente *Estimador de ratings* e *Estimador de ratings* pertencente à fase *offline*. As características de cada componente são as seguintes.

- ***Recomendador***: busca itens na base que possam ser recomendados para o usuário.
- ***Ajustador de pesos***: ajusta o conjunto de pesos para cada usuário, considerando o *feedback* do usuário.
- ***Estimador de ratings***: prediz os *ratings* dos itens resultantes da consulta, considerando os *ratings* implícitos e explícitos dos múltiplos critérios de cada item e o conjunto de pesos do usuário.
- ***Ordenador do ranqueamento***: ranqueia os itens de acordo com *rating* predito pelo componente *Estimador de ratings*,

ordenando-os de acordo com os *rating* preditos. Este componente pode ser implementado por um algoritmo de ordenação, como o *merge sort*.

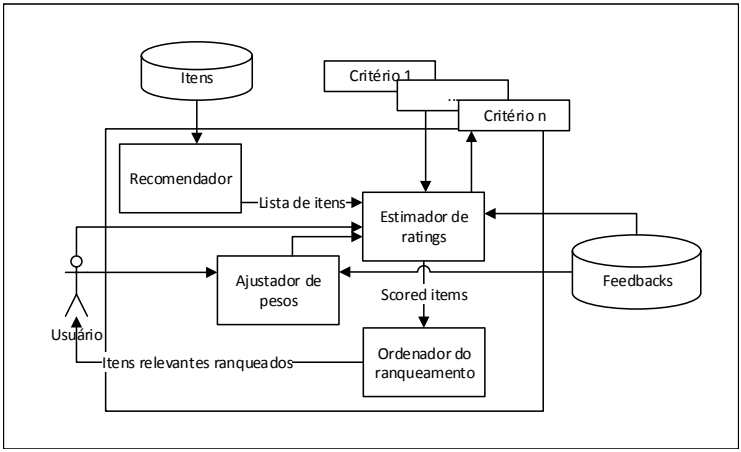

Figura 3 – Visão geral dos componentes da arquitetura

4.2 ESPARSIDADE DOS DADOS

Como discutido anteriormente, o problema da esparsidade de dados pode ser contornado com técnicas de filtragem colaborativa, predizendo-se valores relativos aos *ratings* desconhecidos. Isso, porém, demanda um elevado poder computacional, principalmente pela quantidade de *ratings* inexistentes. Os trabalhos que utilizam filtragem colaborativa para aplicações com múltiplos critérios (NILASHI; IBRAHIM; ITHNIN, 2014; HWANG, 2010; JANNACH; KARAKAYA; GEDIKLI, 2012) predizem os *ratings* para os critérios no intuito de se gerar um ranqueamento de recomendações. Para se ranquear a recomendação de itens é necessário, portanto, predizer um *rating* geral para o item, de forma que quanto maior é o *rating*, mais relevante é o item a determinado usuário. Entretanto, tal *rating* geral, nesses trabalhos, pressupõe que, quando existem *ratings* explícitos desconhecidos, esses serão estimados computacionalmente.

No presente trabalho, é proposto apenas o uso de *ratings* explícitos conhecidos e implícitos, em combinação com seus respectivos pesos,

para a predição do *rating* geral usado para o ranqueamento da recomendação final. A Definição 5 trata da equação proposta para cálculo de um *rating* geral que leva em consideração os *ratings* explícitos conhecidos, *ratings* explícitos desconhecidos, *ratings* implícitos e os pesos dos critérios. Nesta proposta, os *ratings* explícitos conhecidos e os *ratings* implícitos são utilizados no numerador, com os respectivos pesos. Os pesos relacionados aos critérios com *ratings* desconhecidos são somados e usados no denominador da função.

Definição 5 ***Predição do rating geral de um item:*** *Seja W_u o conjunto de $\omega = (w, c)$ em que w é o peso associado ao critério c para o usuário u. Seja também Fb_u o conjunto de feedback para o usuário u. Além disso, $r^{exp}_{c^i,u}$ é o rating $r^{exp} \in (u, r^{exp}, \lambda)$, tal que $(u, r^{exp}, \lambda) \in Fb^{exp}_u$. Analogamente, $r^{imp}_{c^i,u}$ é o rating $r^{imp} \in (u, r^{imp}, \lambda)$, tal que $(u, r^{imp}, \lambda) \in Fb^{imp}_u$. Para ambas definições, λ é um critério c_i para o item i. Então, a fim de se obter a predição de um rating geral $r_{i,u}$ para um item i de um usuário u, é definida a Equação 4.1.*

$$r_{i,u} = \frac{x^k_{i,u} + \sum_{(w,c)\in X_u} w \times r^{imp}_{c^i,u}}{1 - x^{uk}_{i,u}} \tag{4.1}$$

onde $x^k_{i,u}$ representa a soma do produto de pesos com os respectivos ratings conhecidos, como definido na Equação 4.2, enquanto $x^{uk}_{i,u}$ representa a soma dos pesos relacionados aos ratings desconhecidos, como definido na Equação 4.3.

$$x^k_{i,u} = \sum_{(w,c)\in X_u} w \times r^{exp}_{c^i,u} \qquad \forall \quad r^{exp}_{c^i,u} \neq \emptyset \tag{4.2}$$

$$x^{uk}_{i,u} = \sum_{(w,c)\in X_u} w \qquad \forall \quad r^{exp}_{c^i,u} = \emptyset \tag{4.3}$$

Quando um sistema de recomendação, que utilize a abordagem proposta neste trabalho, necessita prover recomendações de itens a determinado usuário, ele deve primeiramente calcular o *rating* mencionado na Definição 5 para todos os itens que deseja recomendar, de modo que sejam preditos os *ratings* gerais de cada item de recomendação e, em seguida, ordená-los de acordo com tais *ratings* calculados.

Exemplificando o uso dessa definição, seja $i = StarWars$ (da Tabela 1) o item que se deseja estimar o *rating* geral para o usuário $u = Bob$. Para o cálculo de $x^{uk}_{i,u}$ toma-se os pesos relacionados com os *ratings* desconhecidos, ou seja, $x^{uk}_{i,u} = 0,05$. Por outro lado, para se obter o

$x^k_{i,u}$, deve-se somar o produto entre os pesos e *ratings* explícitos, ou seja, $x^k_{i,u} = 0,15 \times 0,4 + 0,2 \times 0,8 + 0,10 \times 0,6 = 0,28$. Substituindo-se na fórmula do $r_{i,u}$, tem-se que $r_{i,u} = \frac{0,28+(0,3\times0,8+0,2\times0,9)}{1-0,05} = \frac{0,28+0,42}{0,95} \approx 0,74$.

4.3 AJUSTAMENTO DE PESOS

Nesta seção é proposta a função de maximização, definida em termos matemáticos - através de um problema de otimização -, e em seguida é apresentada a configuração de algoritmo genético que realiza tal otimização. O objetivo da função de maximização e do algoritmo genético é, dado um conjunto de itens com *ratings* conhecidos de determinado usuário, ajustar o conjunto de pesos desse usuário de modo que os *ratings* preditos para tais itens sejam o mais próximo possível dos *ratings* conhecidos. A finalidade desse ajuste é a de se identificar a importância de cada critério para o usuário.

4.3.1 Função de maximização

Considerando que se possua um conjunto de *feedbacks* com *ratings* explícitos de usuários para os itens, o objetivo da função de maximização é ajustar o conjunto de pesos de forma que seja possível estimar o quão próximo desses *ratings* estão os *ratings* computados pela Definição 5, supondo-se utilizar determinado conjunto de pesos. Portanto, a fim de se ajustar o conjunto de pesos que melhor se adeque às preferências do usuário, propõe-se a função de maximização apresentada na Definição 6.

Definição 6 ***Função de maximização****: Seja W_u o conjunto de pesos $\omega = (w, c)$ para o usuário u, em que w é um peso e c é um critério. Seja também Fb^{exp}_u o conjunto de tuplas (u, r^{exp}, λ), tais que $\lambda = i \in I$ e $r^{exp} \neq \emptyset$. Além disso, $r_{i,u}$ é o rating geral predito para o item i, considerando o usuário u, como descrito na Definição 5. Então, a função de maximização é estabelecida na Equação 4.4.*

$$
\begin{aligned}
\underset{X_u}{\textbf{\textit{Maximize}}} \quad & \left(1 - \frac{\sum_{(u,r^{exp},\lambda) \in Fb_u^{exp}} \sqrt{|r^{exp} - r_{i,u}|}}{|Fb_u^{exp}|}\right) \\
\textbf{\textit{subject to}} \quad & \sum_{\omega=(w,c) \in W_u} (w) = 1 \\
& 0 \leq w \leq 1 \ \forall w \in \omega = (w, x) \in W_u.
\end{aligned}
\tag{4.4}
$$

A ideia por trás da Equação 4.4 é ajustar o conjunto de pesos W_u de forma que a diferença entre os *ratings* reais dos itens (*ratings* que o usuário previamente forneceu para os itens) e os *ratings* gerados para esses itens, através da Definição 5, seja mínima. Além disso, usa-se a raiz quadrada do módulo dessa diferença, também conhecido como *Root-mean-square error (RMSE)*. Adota-se também o complemento dessa diferença, uma vez que é apresentado um algoritmo genético que considera parte da Equação 4.4 como a função de *fitness*. Os detalhes são descritos na sub seção 4.3.2.

Por exemplo, supondo-se que se deseje verificar quão bom está o conjunto de pesos da Tabela 1. Supõe-se que os únicos *ratings* fornecidos pelo usuário sejam os da tabela. O *rating*, de acordo com a Definição 5, utilizando o conjunto de pesos, para o filme Star Wars é de 0,74. Já o *rating* computado para o item Filme 2 é de 0,445. O *rating* real fornecido para Star War é de 0,8 e o fornecido para o Filme 2 é de 0,9. Portanto, aplicando-se a Equação 4.4, o resultado seria $1 - \frac{(\sqrt{|0,74-0,8|}+\sqrt{|0,445-0,9|})}{2} = 1 - \frac{(\sqrt{0,06}+\sqrt{0,455})}{2} \approx 1 - \frac{0,24+0,67}{2} \approx 1 - 0,46 \approx 0,54$. Para se determinar o quão bom é o valor predito com relação ao valor real, imagina-se que, quanto mais próximo de zero for o resultado da diferença, melhor é a predição. Como a equação adota o complemento da diferença, então quanto mais próximo o resultado for de 1, mais próximo está o predito do real.

Uma abordagem existente, que é capaz de ajustar um conjunto de peso tal que a diferença entre valores reais e preditos é a regressão múltipla (MONTGOMERY, 2008). Uma regressão linear simples considera um conjunto de pontos bi-dimensionais e ajusta uma linha com inclinação (coeficiente angular) e intercepto (coeficiente linear). Uma regressão múltipla, por outro lado, considera mais variáveis independentes e ajusta outras variáveis além da inclinação e intercepto. Portanto, a fim de se obter o conjunto de pesos desejado, a regressão múltipla poderia ser uma abordagem possível. No entanto, o coeficiente de cada variável em uma regressão múltipla pode ser um número negativo, fa-

zendo com que o conjunto resultante não fosse adequado ao problema, uma vez que pesos não podem ser negativos, de acordo com a Definição 4. Além disso, o conjunto resultante de pesos de uma regressão múltipla não é, necessariamente, normalizado. Portanto, é proposta uma abordagem que utiliza um algoritmo genético para a resolução desse problema de maximização.

4.3.2 Algoritmo genético

O algoritmo genético é baseado no comportamento genético da evolução de indivíduos, em que aqueles mais adaptados tendem a levar adiante seus genes, através de cruzamentos (KOZA, 1992). O algoritmo genético é uma meta-heurística para problemas de otimização, capaz de evoluir determinado conjunto, com base em uma função de otimização (conhecida como função de *fitness*), na finalidade de se otimizar o conjunto. Em um algoritmo genético, quanto maior é o valor que a função de *fitness* resulta, melhor adaptado é o indivíduo. No âmbito do presente trabalho, os indivíduos são os conjuntos de pesos, sendo desejado obter o mais otimizado. Uma vez que o algoritmo genético é uma meta-heurística, é necessário adaptar alguns conceitos para a abordagem proposta: (i) indivíduo (ou código genético) é representado pelo conjunto de pesos, sendo cada peso um cromossomo; (ii) a função de *fitness*, capaz de predizer quão adaptado é algum indivíduo (conjunto de pesos), é a função de maximização definida na Equação 4.4.

Além dos conceitos de indivíduo e função de *fitness*, também é necessário definir duas operações: *crossover* e mutação. As operações de *crossover* e mutação são aquelas que, em cada iteração do algoritmo combinam os códigos genéticos de dois indivíduos e modificam o código genético de um indivíduo. O objetivo da operação de *crossover* é levar adiante cromossomos de indivíduos bem adaptados, buscando valores ótimos. No presente trabalho, são consideradas duas abordagens para implementação da operação de *crossover*, o *crossover* de um ponto (combina dois indivíduos através de um único corte nos códigos genéticos) e o *crossover* de dois pontos (combina dois indivíduos ao se cortar em dois pontos diferentes os códigos genéticos). A operação de mutação, por outro lado, tem como objetivo encontrar valores ótimos globais, evitando-se que determinado ajuste fique preso a ótimos locais, através da mutação aleatória de alguns cromossomos. Um cuidado que se deve ter ao se utilizar a mutação é a taxa com que a mutação vai ocorrer, pois quando ela é alta, a tendência é de que, a cada iteração

do algoritmo genético, conjuntos de pesos sejam aleatórios. Outra observação importante é que, em ambas operações, é possível que sejam criados conjuntos de pesos cuja soma dos elementos seja diferente de 1. Portanto, antes de serem definidas as operações de *crossover* e mutação, é necessário introduzir uma operação de normalização (Equação 4.5), que é capaz de transformar algum conjunto de pesos $W^{(i)}$, cuja soma dos elementos seja diferente de 1, em um conjunto de pesos $W^{(ii)}$ que esteja de acordo com as restrições da função de maximização. A ideia por trás da operação de normalização é dividir cada elemento do conjunto de pesos pela soma dos elementos, de forma que a nova soma dos elementos do conjunto resultante seja igual a 1.

$$W_j^{(ii)} = \frac{W_j^{(i)}}{\sum_{n=1}^{k}(W_n^{(i)})} \tag{4.5}$$

Propõe-se, portanto, que após a aplicação de tais operações, os conjuntos sejam normalizados dividindo-se cada peso do conjunto de pesos pela soma total dos pesos desse conjunto. A operação de *crossover* está definida na Definição 7 enquanto a operação de mutação está definida na Definição 8.

Definição 7 ***Operação de crossover:*** *Seja $W^{(i)}$ e $W^{(ii)}$ dois conjuntos de pesos, tal que $W^{(i)} = \{w_1^{(i)}, \cdots, w_k^{(i)}\}$ e $W^{(ii)} = \{w_1^{(ii)}, \cdots, w_k^{(ii)}\}$. A operação de crossover retorna dois novos conjuntos de pesos, $W^{(iii)}$ e $W^{(iv)}$, que são combinação dos conjuntos $W^{(i)}$ e $W^{(ii)}$ como se segue: $W^{(iii)} = \{w_1^{(i)}, \cdots, w_{\frac{k}{2}}^{(i)}, w_{\frac{k}{2}+1}^{(ii)}, \cdots, w_k^{(ii)}\}$ e $W^{(iv)} = \{w_1^{(ii)}, \cdots, w_{\frac{k}{2}}^{(ii)}, w_{\frac{k}{2}+1}^{(i)}, \cdots, w_k^{(i)}\}$, sendo os conjuntos de pesos resultantes não necessariamente normalizados, há a necessidade de se normalizar o conjunto. Então, seja $w^{(iii)} = \sum w \in W^{(iii)}$ e $w^{(iv)} = \sum w \in W^{(iv)}$. Os conjuntos de pesos normalizados $W^{(iii)'}$ e $W^{(iv)'}$ são:*

$$W^{(iii)'} = \{\frac{w_1^{(i)}}{w^{(iii)}}, \cdots, \frac{w_{\frac{k}{2}}^{(i)}}{w^{(iii)}}, \frac{w_{\frac{k}{2}+1}^{(ii)}}{w^{(iii)}}, \cdots, \frac{w_k^{(ii)}}{w^{(iii)}}\}$$

$$W^{(iv)'} = \{\frac{w_1^{(ii)}}{w^{(iv)}}, \cdots, \frac{w_{\frac{k}{2}}^{(ii)}}{w^{(iv)}}, \frac{w_{\frac{k}{2}+1}^{(i)}}{w^{(iv)}}, \cdots, \frac{w_k^{(i)}}{w^{(iv)}}\}$$

Como $W^{(iii)}$ e $W^{(iv)}$ não são necessariamente normalizados, é necessário normalizar utilizando a Equação 4.5. Para exemplificar o funcionamento dessa operação, sejam os conjuntos de pesos A e B, em que $A = \{0.1, 0.5, 0.2, 0.2\}$ e $B = \{0.4, 0.3, 0.1, 0.2\}$. Primeira-

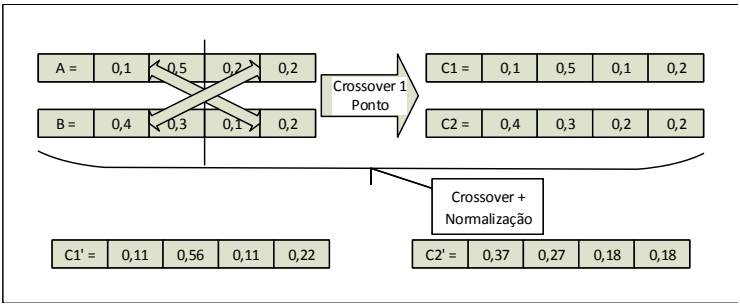

Figura 4 – Exemplo de uma operação de *crossover* com um ponto.

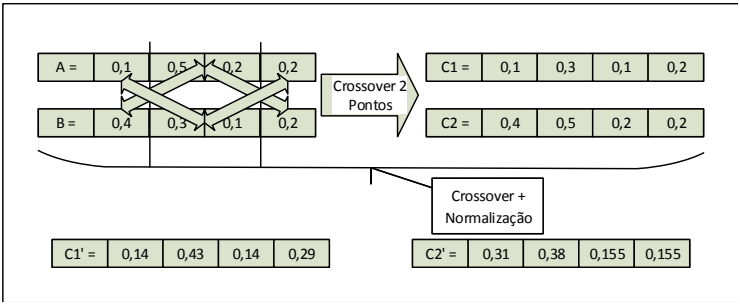

Figura 5 – Exemplo de uma operação de *crossover* com dois pontos.

mente, a operação de *crossover* gera dois novos conjuntos A' e B', sendo $A' = \{0.1, 0.5, 0.1, 0.2\}$ e $B' = \{0.4, 0.3, 0.2, 0.2\}$. Como nenhum conjunto está normalizado (a soma dos elementos de $A' = 0.9$ e a soma dos elementos de $B' = 1.1$), normaliza-se através de Equação 4.5, que divide cada elemento de A' por 0.9 e de B' por 1.1. Os conjuntos resultantes da operação de *crossover* entre A e B são $A' = \{0.11, 0.56, 0.11, 0.22\}$ e $B' = \{0.37, 0.27, 0.18, 0.18\}$. A Figura 4 representa o exemplo destacado. Tal operação é chamada *crossover* de um ponto.

Há outra variação de operação de *crossover* que é usado, que é o *crossover* de dois pontos. Funciona de maneira semelhante, porém os conjuntos são cortados em dois pontos e então reordenados. A Figura 5 representa essa operação com dois pontos para o mesmo conjunto de pesos previamente apresentado.

A operação de mutação, descrita na Definição 8, foca na tentativa

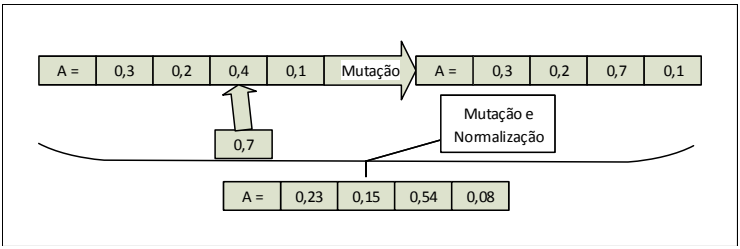

Figura 6 – Exemplo de uma operação de mutação com o terceiro elemento mutacionado.

de se escapar de ótimos locais de modo a se encontrar um ótimo global, através da mutação de algum cromossomo da carga genética.

Definição 8 ***Operação de mutação:*** *Seja* $W^{(i)}$ *um conjunto de pesos, em que* $W^{(i)} = \{w_1^{(i)}, \cdots, w_k^{(i)}\}$. *Seja também* r *um número aleatório no intervalo de* $[0, 1]$, *em que* $r \in \mathbb{R}$, *e seja* j *um número no intervalo de* $[1, k]$, *em que* $j \in \mathbb{Z}$. *A operação de mutação resulta em um conjunto de pesos* $W^{(ii)}$, *em que* $W^{(ii)} = \{w_1^{(ii)}, \cdots, w_k^{(ii)}\}$, *definido como:*

$$w_n^{(ii)} = \begin{cases} r; \ \textit{if } n = j \\ w_n^{(i)}; \ \textit{if } n \neq j \end{cases}$$

Sendo o conjunto $W^{(ii)}$ *não necessariamente normalizado, seja* $w^{(ii)} = \sum w \in W^{(ii)}$. *O conjunto de pesos normalizado* $W^{(ii)'}$ *é:*

$$W^{(ii)'} = \{\frac{w_1^{(ii)}}{w^{(ii)}}, \cdots, \frac{w_k^{(ii)}}{w^{(ii)}}\}$$

Para exemplificar o funcionamento da operação de mutação, seja o conjunto de pesos $A = \{0.3, 0.2, 0.4, 0.1\}$, $j = 3$ e $r = 0.7$, o conjunto resultante antes da normalização é o conjunto $A' = \{0.3, 0.2, 0.7, 0.1\}$, cuja soma dos elementos é 1.3. Após aplicar a normalização através da Equação 4.5, o conjunto resultante é $A' = \{0.23, 0.15, 0.54, 0.08\}$. A Figura 6 ilustra tal operação de mutação.

4.4 ALGORITMO PROPOSTO

Para se efetuar uma recomendação para determinado usuário, propõe-se um algoritmo descrito no pseudo código do Algoritmo 1. Tal algoritmo toma como entrada um conjunto de itens a serem recomendados, um usuário para se recomendar e um número de itens a serem

recomendados. Como resultado, o algoritmo retorna um conjunto de n itens a serem recomendados para o usuário definido. Tal algoritmo faz uma chamada ao algoritmo genético.

Algoritmo 1: Recomendar(I,u,n)

Entradas: Um conjunto de itens I a serem recomendados. O usuário u que receberá a recomendação. Um número n de itens a serem recomendados, tal que $n \leq |I|$.
Saída : Conjunto I_R de n pares (i, r), tal que $i \in I$ e $0 \leq r \leq 1$, a serem recomendados.

1.1 $W_u \leftarrow Maximizar(u, t_c, t_m, p, is)$// `Os valores` t_c, t_m, p `e` is `são parâmetros para o algoritmo genético.`;
1.2 $I_R \leftarrow \{\emptyset\}$;
1.3 **for** $i \in I$ **do**
1.4 | $r \leftarrow r_{i,u}$ *de acordo com a Definição 5*;
1.5 | $I_R \leftarrow I_R \cup \{(i, r)\}$;
1.6 **end**
1.7 $I_R \leftarrow$ *Ordenar* I_R *de forma decrescente, de acordo com* r;
1.8 *Retorna* $\{(i, r)_1, ..., (i, r)_n\} \subseteq I_R$;

O algoritmo genético proposto no pseudo código, formalizado no Algoritmo 2, toma como entrada um usuário, taxa de *crossover*, taxa de mutação, tamanho da população e quantidade de iterações. Tal algoritmo tem, como resultado, um conjunto de pesos otimizado para o usuário. Esse algoritmo, por sua vez, faz chamada aos Algoritmos 3 e 4 que, respectivamente, computam os casais selecionados do algoritmo genético, de acordo com o método da roleta (KOZA, 1992), e geram a próxima geração, de acordo com as operações de *crossover* e mutação.

O algoritmo genético proposto utiliza o método da roleta (KOZA, 1992) para gerar a geração da próxima iteração do algoritmo genético. O algoritmo da roleta pode ser identificado no Algoritmo 3. Tal algoritmo toma como entrada uma população e, com base nos valores de *fitness* de cada indivíduo na população, gera diversos pares de indivíduos que são utilizados para as operações de *crossover* e mutação.

Com os casais gerados através do método da roleta, a próxima geração do algoritmo genético é computada, como verificado no Algoritmo 4. Essa população será resultado de operações de *crossover* e mutação. Tais operações ocorrem em uma taxa enviada por parâmetro.

O procedimento de gerar casais e aplicar as operações de mutação e *crossover* é repetida até um número máximo de iterações previamente definida. A cada iteração do algoritmo, é possível que se identifique um

Algoritmo 2: Maximizar(u, t_c, t_m, p, is)

Entrada: Um usuário u. Taxa de *crossover* t_c. Taxa de mutação t_m. Quantidade populacional p. Número de iterações is do algoritmo genético.

Saída : Conjunto $W_{u,opt}$ de pesos, tal que W_u seja otimizado.

2.1 $C_W \leftarrow \{\emptyset\}$;
2.2 **for** $i \leftarrow 1$ **to** p **do**
2.3 $\quad W_i \leftarrow$ *Conjunto aleatório de pesos normalizados*;
2.4 $\quad C_W \leftarrow C_W \cup \{W_i\}$ `// Inicializando primeira população do AG` ;
2.5 **end**
2.6 $Roleta \leftarrow \{\emptyset\}$`// Roleta é um conjunto de triplas` (W, a, b) `em que` W `é o conjunto de pesos resultante para um` k `entre` a `e` b;
2.7 $S \leftarrow 0$;
2.8 $W_{u,opt} \leftarrow \emptyset$;
2.9 $W_{u,opt}.r \leftarrow 0$;
2.10 **for** $i \leftarrow 1$ **to** is **do**
2.11 $\quad$ **for** $j \leftarrow 1$ **to** p **do**
2.12 $\quad\quad r_j \leftarrow$ *computar valor de fitness de* $W_j \in C_W$, de acordo com Definição 4.4 ;
2.13 $\quad\quad S \leftarrow S + r_j$;
2.14 $\quad\quad Roleta \leftarrow Roleta \cup \{(W_j, S - r_j, S)\}$;
2.15 $\quad\quad$ **if** $r_j > W_{u,opt}.r$ **then**
2.16 $\quad\quad\quad W_{u,opt} \leftarrow W_j$;
2.17 $\quad\quad\quad W_{u,opt}.r \leftarrow r_j$;
2.18 $\quad\quad$ **end**
2.19 $\quad$ **end**
2.20 $\quad Casais \leftarrow ComputaCasais(Roleta, p)$;
2.21 $\quad ProxGeracao \leftarrow ComputaProximaGeracao(Casais, t_c, t_m)$;
2.22 $\quad C_W \leftarrow ProxGeracao$;
2.23 **end**
2.24 *Retorna* $W_{u,opt}$;

Algoritmo 3: ComputaCasais(*Roleta*,*p*)

Entradas: Uma *Roleta*, contendo um conjunto de triplas (W, a, b) e p, o tamanho da população
Saída : Conjunto *Casais* de pares W_1, W_2 selecionados aleatoriamente da *Roleta*.

```
3.1 Casais ← {∅};
3.2 for j ← 1 to p/2 do
3.3 |   r1 ← aleatório;
3.4 |   r2 ← aleatório;
3.5 |   W1 ← conjunto W de Roleta, tal que a ≤ r1 ≤ b;
3.6 |   W2 ← conjunto W de Roleta, tal que a ≤ r2 ≤ b;
3.7 |   Casais ← Casais ∪ {(W1, W2)};
3.8 end
3.9 Retorna Casais;
```

Algoritmo 4: ComputaProximaGeracao(*Casais*,t_c,t_m)

Entradas: Um conjunto *Casais* de pares (W_1, W_2), uma taxa de *crossover* t_c e uma taxa de mutação t_m
Saída : Conjunto *ProxGeracao* de W resultante das junções de *Casais*.

```
4.1  ProxGeracao ← {∅};
4.2  for (W1, W2) ∈ Casal do
4.3  |   t ← aleatório;
4.4  |   if t < tc then
4.5  |   |   (W1, W2) ← Crossover(W1, W2);
4.6  |   |   ProxGeracao ← ProxGeracao ∪ {(W1), (W2)};
4.7  |   end
4.8  |   else
4.9  |   |   ProxGeracao ← ProxGeracao ∪ {(W1), (W2)};
4.10 |   end
4.11 end
4.12 for W ∈ ProxGeracao do
4.13 |   for wi ∈ W do
4.14 |   |   t ← aleatório;
4.15 |   |   if t < tm then
4.16 |   |   |   wi ← aleatório;
4.17 |   |   end
4.18 |   end
4.19 |   W ← normalizar(W);
4.20 end
4.21 Retorna ProxGeracao;
```

novo conjunto de pesos que melhor satisfaça a função de maximização definida na Definição 4.4. Com base nesse conjunto de pesos, são computados os *ratings* dos itens a serem recomendados.

5 AVALIAÇÃO EXPERIMENTAL

Neste capítulo são apresentados os experimentos realizados com o intuito de validar a abordagem proposta. O principal objetivo dos experimentos é demostrar que o algoritmo genético, através da função de *fitness* proposta, é capaz de produzir uma estratégia de pesos que melhora a precisão dos resultados, mesmo que os experimentos tenham sido realizados em um conjunto de dados bastante esparso.

5.1 *OVERVIEW* DO PROCESSO EXPERIMENTAL

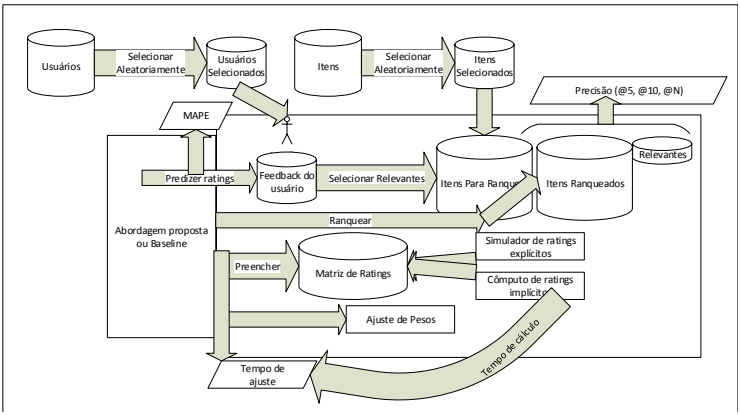

Figura 7 – *Overview* do processo experimental.

O processo experimental está ilustrado na Figura 7. Primeiramente selecionam-se aleatoriamente os usuários que farão parte do processo experimental. Também são selecionados itens que deverão ser recomendados aos usuários. Para cada usuário do processo experimental, adicionam-se os itens relevantes aos itens selecionados anteriormente. Esses itens selecionados são ranqueados, de acordo com cada abordagem e, então, comparam-se os itens ranqueados com os itens relevantes para se obter a precisão para a abordagem utilizada.

Simultaneamente computa-se o tempo para preenchimento da matriz de *ratings* e para o ajuste de peso, bem como o cômputo dos *ratings* implícitos. Esse tempo é contabilizado como tempo de ajuste

(*tempo offline*). De acordo com cada abordagem são preditos os *ratings* para cada item do *feedback* e então comparam-se os *ratings* preditos com os *ratings* fornecidos para se obter o MAPE.

Para o preenchimento da matriz de *ratings*, são utilizadas as técnicas de filtragem colaborativa propostas pelo *baseline* para cálculo dos *ratings* explícitos. Para o cálculo dos *ratings* implícitos são utilizadas as abordagens de cálculo de *rating* implícito propostas na seção 5.4.1. Uma vez que o *MovieLens* não possui *ratings* de critéros explícitos, tais *ratings* são simulados.

5.2 BASE DE DADOS

Para os experimentos, foi usada a base de dados do Movielens Latest Database[1], uma vez que essa base é largamente utilizada na comunidade de recuperação de informação (ZHANG et al., 2014a; MENG; SHAO et al., 2014; LI; YANG; ZHANG, 2013), além de conter um grande número de *ratings*. No total, o Movielens contém 22 milhões de *ratings* para 33 mil filmes, aplicados por 240 mil usuários, sendo que cada *rating* é dado em uma escala de 1 a 5 estrelas para os filmes. Para trabalhar com valores numéricos, tais *ratings* foram normalizados divindo-os por 5. Como a base de dados do MovieLens não contém *ratings* explícitos para os critérios, e sim para o item como um todo, então foi necessário simulá-los conforme discutido mais adiante na Seção 5.4. Tal simulação resultou em um conjunto de 5 critérios para cada um dos filmes: `enredo`, `atuação` e `qualidade`, que possuem *ratings* explícitos, e `data de lançamento` e `gênero` que possuem *ratings* implícitos. Como essa base de dados possui um total de 22 milhões de avaliações fornecidas e existem aproximadamente 7,92 bilhões de pares de usuário e filmes (33 mil filmes e 240 mil usuários), existem aproximadamente 7,898 bilhões de avaliações desconhecidas, e a esparsidade dessa base de dados é de aproximadamente 99,7% ($\frac{7{,}898}{7.92}$).

5.3 BASELINES E PARÂMETROS DE CONFIGURAÇÃO

Com o intuito de comparar a abordagem proposta com outros trabalhos propostos na literatura, foram considerados quatro *baselines*. Dois deles levam em conta a proposta de Jannach (JANNACH; KARAKAYA; GEDIKLI, 2012), outro considera o proposto por Nilashi (NILASHI;

[1] http://grouplens.org/datasets/movielens/

IBRAHIM; ITHNIN, 2014) e outro o trabalho de Hwang (HWANG, 2010). Os *baselines* foram configurados da seguinte forma:

- (i) Jannach com predição de *rating* (JwR) e sem predição de *rating* (JnR), usando o algoritmo SV-regress proposto com os parâmetros descritos pelos autores;
- (ii) Nilashi (Nil), com as regras *fuzzy* manualmente definidas;
- (iii) Hwang (HwG), com os parâmetros e funções definidos pelo autor.

Além desses *baselines*, foram adotadas duas variações para o trabalho proposto: uma delas utilizando *crossover* de um ponto (GW-1P) e a outra usando *crossover* de 2 pontos (GW-2P). Os parâmetros utilizados para o algoritmo genético foram os seguintes: taxa de *crossover* foi definida como $0,7$, a taxa de mutação foi $0,01$, o tamanho da população foi 50 e foram consideradas 1 milhão de iterações para se obter a otimização da função de *fitness*.

5.4 METODOLOGIA UTILIZADA

Os experimentos foram conduzidos através da ordenação dos itens de recomendação utilizando cada uma das abordagens apontadas na Seção 5.3, fazendo uso do mesmo conjunto de dados (mesmos usuários e mesmos itens).

Os experimentos foram repetidos para 100 usuários aleatoriamente selecionados e foram considerados os seguintes critérios: `enredo`, `atuação`, `qualidade`, `data de lançamento` e `gênero`. Como já mencionado, a base de dados do Movielens não contém *ratings* explícitos para os critérios, então foi necessário simulá-los.

A simulação foi feita da seguinte forma: para cada usuário, selecionou-se os itens que faziam parte do *feedback*. Então, para cada critério com *rating* explícito, simulou-se uma nota para o critério. Como nem sempre um usuário fornece *ratings* aos critérios de um item, foi utilizada uma taxa de 60% para cada critério que, se tal taxa fosse acionada, seria fornecido um *rating* ao critério. O *rating* do critério é o *rating* do item somado com um valor aleatório no intervalo de $-0,4$ a $0,4$, respeitando-se a restrição de que o *rating* deve estar no intervalo entre 0 e 1. Para se exemplificar essa simulação, seja $0,6$ o *rating* fornecido para o filme A. Para cada critério com *rating* explícito (`atuação`, `qualidade` e `enredo`) há 60% de chance de o usuário fornecer *rating* (simulado) para tal item. Ao se fazer essa simulação para o item A, e com

a suposição de que apenas o critério `enredo` tenha *ratings* simulado (ou seja, os *ratings* de `atuação` e `qualidade` se mantém desconhecidos). Seja também $-0,1$ o número que foi aleatoriamente escolhido entre $[-0,4, 0,4]$. Então, o *rating* ligado ao critério `enredo` do filme A é de $0,5$ $(0,6 - 0,1)$ na simulação feita.

Os experimentos foram realizados de forma a mensurar as quatro situações a seguir.

- **(i) Precisão na recomendação de itens:** neste caso, simulou-se um cenário de recomendação de filmes para usuários. Para tal simulação, foi selecionado um conjunto de itens aleatórios e, nesse conjunto, foram adicionados os filmes relevantes para o usuário. Então, para cada item desse conjunto, foram preditos *ratings* gerais para cada item, de acordo com a Definição 5. Os itens foram ranqueados e, com base nos itens do topo do ranqueamento, foram mensuradas a precisão e cobertura do ranqueamento;

- **(ii) Efeitos da esparsidade de dados:** nesta situação, o conjunto de dados utilizado para o experimento possui pelo menos 20 *ratings* por usuário. Simulou-se, então, um cenário em que 95% desses *feedbacks* foram removidos da base, fazendo com que o conjunto de *ratings* conhecidos ficasse ainda mais esparso. Dessa maneira, é possível avaliar os efeitos de uma base de dados mais esparsa para a abordagem proposta;

- **(iii) Tempo necessário para recomendação:** neste caso, foram avaliados os tempos necessários para se aplicar as abordagens testadas, verificando-se a viabilidade de cada uma em sistemas Web. Para tais experimentos, foram considerados os tempos *online* e *offline*. No primeiro caso, foi medido o tempo necessário para ranqueamento dos itens para recomendação, enquanto no segundo caso, foi considerado o tempo necessário para a configuração dos dados para recomendação. Por configuração dos dados entende-se os seguintes passos: predição de *ratings* implícitos, predição de *ratings* explícitos (para as abordagens que necessitam dessa predição) e ajuste de pesos (para abordagens que necessitam de tal ajuste);

- **(iv) Diferença entre *ratings* preditos com os reais:** estimou-se *ratings* para todos os itens com *ratings* conhecidos, de forma a se verificar a diferença entre o real, fornecido pelo usuário, e o predito, computado pelo sistema.

Os resultados dos experimentos foram medidos em termos de precisão, tempo (em milissegundos) e erro percentual absoluto médio (MAPE). Para isso, foram consideradas os top 5 (P@5), top 10 (P@10) e top N (P@N), onde N significa o número de itens relevantes (por exemplo, no caso de existirem 7 itens relevantes para um usuário, são considerados os top 7 resultados). Um item é considerado relevante quando possui um *rating* maior ou igual a $0,6$ (ou seja, no mínimo 3 estrelas). Todos os experimentos foram feitos em um conjunto de 150 itens aleatórios, além dos itens relevantes, ranqueados de acordo com o *rating* geral predito para cada item. *Time Off* significa o tempo necessário para o processamento *offline*, enquanto *Time On* significa o tempo para recomendação dos resultados.

5.4.1 Cálculo dos *ratings* implícitos

Como os critérios `data de lançamento` e `gênero` dos filmes do Movielens são relacionados a *ratings* implícitos, é preciso gerar tais *ratings*. Para o critério `data de lançamento` proposto, tem-se como hipótese de que os usuários preferem filmes mais novos. Então, a ideia é calcular *ratings* maiores para novos filmes. Como o filme mais antigo da base de dados é datado no ano de 1906, o *rating* para um filme é calculado de acordo com a Definição 9.

Definição 9 ***Rating para critério de data de lançamento mais recente:*** *Seja y o ano atual, i um item, em que i_d representa a data de lançamento dele. Então, o rating para o critério de data mais recente é computado para determinado usuário u como sendo o seguinte:* $r^{imp(i,u)} = max\{0, \frac{(i_d - 1906)}{(y - 1906)}\}$.

A intuição do cálculo do *rating* para o critério `data de lançamento` proposta é dar maiores *ratings* para filmes mais recentes. Utiliza-se o ano do filme mais antigo registrado (1906) como fator para dar maior destaque às diferenças dos anos dos filmes. Dessa forma, sendo o ano atual 2016, por exemplo, um filme de 1930 tem *rating* de 0,22, enquanto um filme de 2010 tem *rating* de 0,95.

O outro critério para o qual é necessário gerar um *rating* implícito é `gênero`. O *rating* de gênero deve ser diferente para diferentes usuários, uma vez que há usuários que gostam de um determinado gênero em específico enquanto outros usuários não. O *rating* de gênero é calculado em duas partes: (i) os níveis de interesse de cada gênero para o usuário, que indica dentre os itens que o usuário previamente forneceu *rating*,

Tabela 3 – Resultados dos experimentos considerando todos os *feedbacks*

	Todos os feedbacks					
Method	P@5	P@10	P@N	T-Off(ms)	T-On(ms)	MAPE (%)
GW-1P	0.722	**0.611***	0.736	**31040***	**42.6***	**12.84***
GW-2P	0.678	0.600	0.758	35195	47.5	13.82
Nil	0.592	0.516	0.656	78583	110.8	17.89
HwG	0.594	0.485	0.581	95909	135.3	22.80
JnR	0.590	0.523	0.647	45275	63.0	15.65
JwR	**0.728***	0.599	**0.759***	115723	159.6	

quais deles contém um determinado gênero (Definição 10); e (ii) *rating* para o critério `gênero`, baseado nos gêneros do filme e nos níveis de interesse de cada gênero para o usuário (Definição 11).

Definição 10 ***Nível de interesse do gênero:*** *Seja* $G = \{g_1, \cdots, g_m\}$ *um conjunto de* m *gêneros e* $G_i = \{g_{i,1}, \cdots, g_{i,j}\} \subseteq G$ *o conjunto dos gêneros de um item* i*. Cada gênero* $g \in G$ *possui um rating* $r_{g,u}$ *associado a um usuário* u*, computado pela seguinte equação:* $r_{g,u} = \frac{\sum_{(u,r,\lambda) \in Fb^{exp}_{g,u}}(r)}{|Fb^{exp}_{g,u}|}$*, em que* $Fb^{exp}_{g,u} = (u, r, \lambda) | \lambda \in I \wedge g \in G_\lambda \subseteq Fb^{exp}$ *é o conjunto de feedback explícito do usuário* u*, cujos itens contenham o gênero* g*.*

Definição 11 ***Rating de gênero:*** *Seja* i *um item que contém um conjunto de gêneros* $G_i = \{g_{i,1}, \cdots, g_{i,j}\} \subseteq G$*. Seja também* $r_{g,u}$ *o nível de interesse de um gênero* g *para o usuário* u*, calculado através da Definição 10, então o rating implícito do gênero é calculado através da seguinte equação:* $r^{imp(i,u)} = \frac{\sum_{g_j \in G_i}(r_{g_j,u})}{|G_i|}$*.*

Para o cálculo do *rating* para o critério de `gênero`, é necessário primeiramente calcular o nível de interesse, que indica, dentre os itens que o usuário previamente forneceu *rating*, quais deles contém esse gênero. A média desses *ratings* é o nível de interesse do gênero. Sabendo-se o nível de interesse de todos os gêneros para determinado usuário, calcular o *rating* para o critério de `gênero` de um item fica trivial, sendo tal *rating* a média dos interesses dos gêneros do item.

Tabela 4 – Resultados dos experimentos considerando poucos *feedbacks*

	Poucos feedbacks					
Method	P@5	P@10	P@N	T-Off(ms)	T-On(ms)	MAPE (%)
GW-1P	**0.228***	**0.182***	**0.231***	**29080***	48.3	**27.96***
GW-2P	0.188	0.161	0.194	31577	**48.2***	29.02
Nil	0.170	0.148	0.190	82529	48.5	35.37
HwG	0.146	0.122	0.162	101882	49.0	38.88
JnR	0.164	0.132	0.152	38109	49.0	31.62
JwR	0.172	0.139	0.182	121463	47.6	

5.5 RESULTADOS DOS EXPERIMENTOS E DISCUSSÕES

Esta seção mostra os resultados dos experimentos em dois cenários. O primeiro deles considera todos os *feedbacks* dos usuários presentes na base de dados do Movielens e o resultado está disposto na Tabela 3. O segundo considera apenas 5% desses **feedbacks** e o resultado está disposto na Tabela 4. Em ambas as tabelas são mostrados os resultados, de forma resumida, sobre a precisão (P@5, P@10 e P@N), tempo de ajuste (T-Off) em milissegundos, tempo de recomendação (T-On) e MAPE (em %). Nas próximas subseções são discutidos detalhamente os resultados de acordo com cada métrica especificada. Nos gráficos mostrados nas subseções que se seguem, o Cenário 1 representa o cenário com todos os *feedbacks*, enquanto o Cenário 2 representa o cenário com apenas 5% dos *feedbacks*.

5.5.1 Discussão sobre a precisão

O gráfico mostrado na Figura 8 representa o disposto nas Tabelas 3 e 4 para a precisão das diferentes abordagens. Tal gráfico dispõe as precisões para os *top 5*, *top 10* e *top n* resultados nos dois cenários. Ao se analisar a Tabela 3, nota-se que a precisão nos *top 5* e *top n* é maior na abordagem de Jannach (JANNACH; KARAKAYA; GEDIKLI, 2012), enquanto nos *top 10*, a abordagem proposta no presente trabalho foi superior que as demais abordagens. Contudo, ao se analisar graficamente o resultado, nota-se que tais diferenças são pequenas, de modo que é possível se dizer que a abordagem proposta no presente trabalho é, ao menos, tão boa quanto o trabalho de Jannach.

Todavia, ao se analisar o cenário mais esparso, nota-se que os maiores valores de precisão estão concentrados na abordagem GW-1P,

ou seja, na abordagem proposta no presente trabalho, sendo o algoritmo genético configurado com *crossover* de 1 ponto. Graficamente, nota-se que a diferença entre o proposto neste trabalho e o proposto pelos demais trabalhos cresce e a abordagem de Jannach, que no primeiro cenário se mostra superior aos demais *baselines*, passa a ter a precisão mais próxima dos demais *baselines* no cenário 2.

Tais resultados indicam que o problema de esparsidade de dados foi mitigado ao se aplicar a abordagem proposta pelo presente trabalho. É importante notar, contudo, que apesar de mitigado, o problema afetou o desempenho da abordagem proposta, visto que a precisão dos resultados se tornou menor no cenário 2.

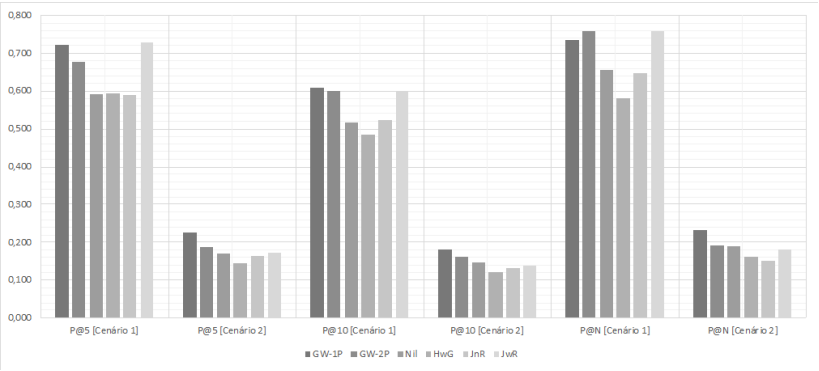

Figura 8 – Gráfico da precisão

5.5.2 Discussão sobre o tempo

Para se analisar o tempo, é necessário entender a diferença entre o significado do *T-On* e do *T-Off*. O primeiro se refere ao tempo que o sistema de recomendação leva para, dados os ajustes necessários, ordenar os itens a serem recomendados e recomendá-los. Em termos práticos, é o tempo entre o usuário de um sistema clicar em um botão e ter a resposta de volta. Nota-se, pelas Tabelas 3 e 4 que, em média, o maior tempo de resposta foi de 159ms, o que indica que todas as abordagens possuem um bom tempo de resposta.

Contudo, quando se trata de ajuste, o tempo de processamento aumenta. Isso porque, em abordagens que necessitam de filtragem cola-

borativa para preencher os *ratings* desconhecidos, é necessário estimar a maioria dos *ratings*. Como as bases de dados em sistemas de recomendação são esparsas, então esse tempo de ajuste tende a ser grande. No caso deste trabalho, o tempo de ajuste é o de se preencher os *ratings* implícitos e, também, de serem ajustados os pesos para os usuários. O gráfico ilustrado na Figura 9 dispõe o tempo necessário para serem feitos os ajustes necessários nos dois cenários.

Nota-se que em abordagens que utilizam filtragem colaborativa para prever *ratings* desconhecidos, o tempo é maior que as demais, o que indica que predizer apenas *ratings* implícitos é uma boa abordagem. Além disso, ao se analisar o cenário 1 e o cenário 2, nota-se que as abordagens que precisam predizer *ratings* desconhecidos possuem um tempo maior de ajuste no cenário mais esparso. Isso ocorre pelo fato de que, quanto mais esparso for o cenário, mais *ratings* desconhecidos existem e, portanto, mais tempo será gasto para predizê-los. Por outro lado, as abordagens que apenas computam *ratings* implícitos tem o tempo de ajuste diminuído em um cenário mais esparso. Isso acontece porque, ao se ter menos dados conhecidos, o ajuste do algoritmo genético leva menos tempo, uma vez que há menos *ratings* conhecidos. Com menos *ratings* conhecidos, a função de *fitness* precisa predizer menos *ratings*. Portanto, com as análises acerca do tempo, nota-se que é necessário menos tempo para o sistema de recomendação ser ajustado, ao se adotar uma estratégia que busque não predizer todos os *ratings* desconhecidos.

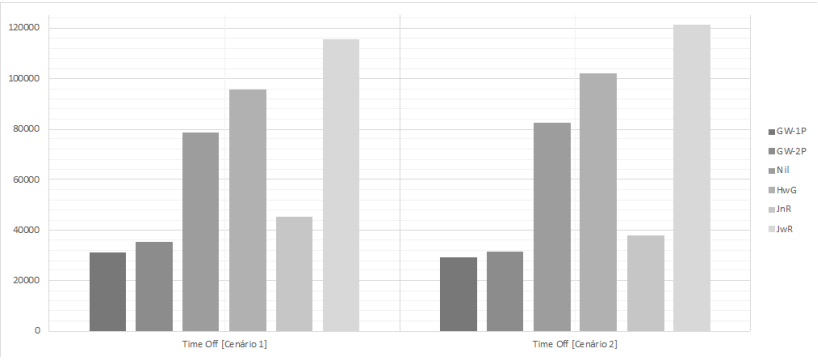

Figura 9 – Gráfico do tempo

5.5.3 Discussão sobre o MAPE

O *Mean Average Percentage Error (MAPE)* é uma métrica que avalia quão boa está determinada predição. Quanto maior é o MAPE, pior é a predição. Uma predição perfeita terá MAPE de 0%. Para se analisar o MAPE, é possível analisar graficamente através da Figura 10, bem como analisar numericamente através das Tabelas 3 e 4.

O algoritmo genético proposto no presente trabalho maximiza uma função que tem como objetivo fazer com que os *ratings* preditos sejam o mais próximo possível dos *ratings* reais. Portanto, o esperado é que o MAPE da abordagem proposta tenha um valor pequeno. Ao ser analisado o MAPE, nota-se que ambas propostas de algoritmo genético tiveram um resultado melhor que os demais *baselines*. Outra observação importante é que, ao mudar o cenário para mais esparso, o MAPE aumenta consideravelmente. Isso ocorre pelo fato de o algoritmo genético maximizar uma função que possui poucas variáveis. Contudo, mesmo no cenário mais esparso, o MAPE ainda foi maior no algoritmo genético proposto no presente trabalho. Com isso exposto, é evidenciado que o algoritmo genético proposto de fato resulta em um bom ajuste.

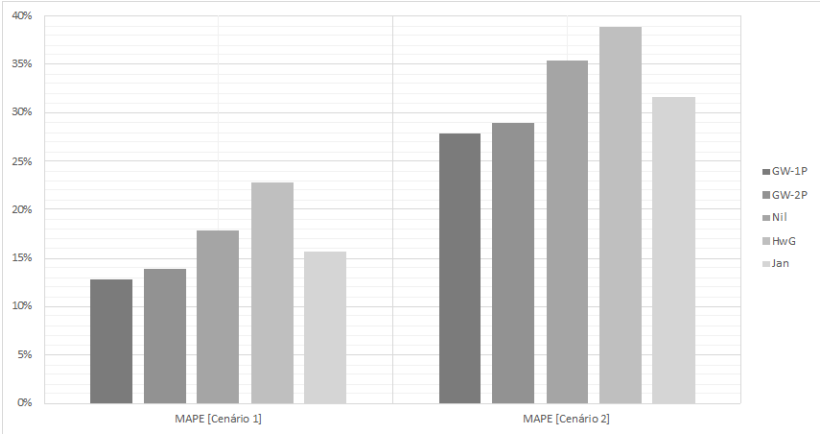

Figura 10 – Gráfico do MAPE

5.5.4 Testes de hipóteses

No intuito de se validar os testes, fez-se testes de hipóteses. Tais testes identificam se as variações encontradas nos resultados são significantes ou não. Utilizou-se teste de médias pareado para as validações dos resultados (MONTGOMERY, 2008). Para se fazer testes pareados, é necessário comparar os resultados usados para uma mesma amostra usando abordagens diferentes. O experimento utilizou um conjunto de 100 usuários, aplicando-se cada uma das abordagens do *baseline* para o mesmo conjunto, o que possibilita o uso de teste pareado. Em testes pareados, é calculado um fator chamado p-valor que, quanto menor, maior é a probabilidade de os resultados possuírem diferença significante.

Para se realizar o teste de média, primeiro adotam-se duas hipóteses: uma hipótese H_0, que é a hipótese de que não existe diferença entre as médias, e uma hipótese H_1, que é a hipótese de que existe diferença entre as médias. No teste adotado, usou-se nível de significância de 5%, ou seja, se o p-valor for menor que 5%, a hipótese H_0 é rejeitada e, portanto, é provado estatisticamente que há diferenças significantes para as médias. Todos os testes de hipóteses foram feitos utilizando o *software R-Project*[2].

Os testes pareados foram feitos baseando-se sempre na melhor abordagem do *baseline* com a versão de *crossover* de um ponto (GW-1P) proposta. As tabelas 5 e 6 mostram o resultado dos testes de hipóteses. As colunas indicam a métrica que está sendo verificada e as linhas indicam, respectivamente, a hipótese nula (H_0), hipótese alternativa (H_1), p-valor e a conclusão.

Através do teste de hipóteses, nota-se que, em um cenário com todos os *feedbacks*, não existe diferença significante entre a precisão da abordagem de (JANNACH; KARAKAYA; GEDIKLI, 2012) com a abordagem proposta. O que indica que, mesmo que numericamente as precisões de tal abordagem sejam superiores à proposta neste trabalho, estatisticamente não há evidências de que sejam significantemente superiores. Por outro lado, em um cenário com poucos *feedbacks*, as precisões obtidas pela abordagem proposta neste trabalho nos 5 primeiros e nos 10 primeiros resultados (P@5 e P@10, respectivamente) mostra-se superior, até mesmo estatisticamente ao nível de 5%, do que as melhores abordagens do *baseline*. Analisando-se o teste feito do Tempo de ajuste (T-off) e do MAPE, nota-se que a abordagem proposta neste

[2] `https://www.r-project.org/`

Tabela 5 – Teste de hipóteses ao nível de significância de 5% para todos os *feedbacks*

Teste de Hipóteses		P@5	P@10	P@N	T-Off	MAPE
Hipóteses	H_0	JwR = GW-1P	GW-1P = JwR	JwR = GW-1P	GW-1P = JnR	GW-1P = JnR
	H_1	JwR >GW-1P	GW-1P >JwR	JwR >GW-1P	GW-1P <JnR	GW-1P <JnR
	p-valor	0,4049	0,2348	0,09545	<2.2*10^(-16)	<2.2*10^(-16)
	Conclusão	Aceita H_0	Aceita H_0	Aceita H_0	Rejeita H_0	Rejeita H_0

Tabela 6 – Teste de hipóteses ao nível de significância de 5% para poucos *feedbacks*

Teste de Hipóteses		P@5	P@10	P@N	T-Off	MAPE
Hipóteses	H_0	GW-1P = JwR	GW-1P = Nil	GW-1P = Nil	GW-1P = JnR	GW-1P = JnR
	H_1	GW-1P >JwR	GW-1P >Nil	GW-1P >Nil	GW-1P <JnR	GW-1P <JnR
	p-valor	0,01402	0,01307	0,0685	<2.2*10^(-16)	1.54e-15
	Conclusão	Rejeita H_0	Rejeita H_0	Aceita H_0	Rejeita H_0	Rejeita H_0

trabalho realmente possui diferença significativa, o que valida a hipótese de que a abordagem proposta reduz o tempo de processamento *offline*.

5.5.5 Considerações finais do resultado

Comparando-se primeiramente as duas propostas deste trabalho (duas primeiras linhas das tabelas 3 e 4), quando considerado o cenário com todos os *feedbacks*, os experimentos mostram que o ajuste de pesos utilizando *crossover* de 2 pontos supera o ajuste do *crossover* de 1 ponto em termos de P@N, embora tenha um resultado pior em termos de P@5 e P@10. O *crossover* de 2 pontos, nesse cenário, tem a desvantagem de demandar 13% a mais de tempo de processamento *offline*.

Para as demais abordagens do *baseline*, o tempo gasto para ajustes de predição de *ratings* desconhecidos foi muito maior que as duas variações da abordagem proposta neste trabalho. Comparando-se o tempo da variação *crossover* de 1 ponto, a abordagem de Nilashi (NILASHI; IBRAHIM; ITHNIN, 2014) teve um tempo 153% maior, enquanto a abordagem de Jannach (JANNACH; KARAKAYA; GEDIKLI, 2012) excedeu 272%. A diferença entre os tempos é, como mostrado na Seção 5.5.4, significativa.

Apesar de mais lenta em termos de tempo para ajustamento, a abordagem de Jannach mostrou ser levemente melhor que a abordagem proposta neste trabalho nos resultados considerando os top 5 e os top N resultados em um cenário com todos os *feedbacks*. Isso mostra que a predição de *ratings* desconhecidos pode aumentar levemente a precisão nesse cenário. Contudo, tais diferenças não são significantes do ponto de vista estatístico, como mostrado na Seção 5.5.4.

Ao se mudar de cenário, reduzindo 95% do *feedback* de todos os usuários, deixando a base de dados de *ratings* ainda mais esparsa, as duas variações da abordagem proposta neste trabalho superaram os demais *baselines*, inclusive do ponto de vista estatístico no caso de precisão dos 5 primeiros e 10 primeiros resultados. Nesse cenário, pelo fato de as predições de *ratings* desconhecidos se tornarem menos confiáveis, as abordagens que necessitam predizer *ratings* desconhecidos, além de levar mais tempo com essa predição, tendem a perder precisão. Outro ponto interessante no cenário de poucos *feedbacks* é o fato de o tempo necessário para ajuste diminui na abordagem proposta, pelo fato de se ter menos informações para ajuste, mas aumenta nas demais, pelos motivos já descritos.

Considerando predição de *ratings*, a abordagem proposta mostrou melhor desempenho que as *baselines*. O erro médio da abordagem proposta (com *crossover* de um ponto), em um cenário menos esparso, manteve uma média de erro de $12,84\%$, ao passo que tal erro subiu para $27,96\%$ em um cenário mais esparso. Nos dois cenários, o *baseline* que melhor manteve desempenho foi a abordagem de Jannach (JANNACH; KARAKAYA; GEDIKLI, 2012), cujo MAPE foi de $15,65\%$ e $31,62\%$, respectivamente.

6 CONCLUSÕES E TRABALHOS FUTUROS

Os experimentos acerca da abordagem proposta mostram que o algoritmo genético proposto é capaz de maximizar corretamente a função. Isso é evidenciado através da métrica de *Mean Average Percentage Error (MAPE)*. O MAPE para o presente trabalho se mostrou melhor que para os demais *baselines* utilizados. Além disso, a precisão das recomendações feitas através do proposto neste trabalho se mostrou, no pior dos casos, ao menos tão bom quanto as recomendações feitas pelo melhor dos *baselines*.

Outro problema identificado no trabalho é que, em muitos dos trabalhos que recomendam itens baseado em múltiplos critérios, há a necessidade de serem preditos *ratings* para os critérios com *ratings* desconhecidos. Isso se torna um problema em bases de dados esparsas, visto que é demandado muito tempo para que sejam preditos todos os *ratings* desconhecidos. Além disso, a predição é feita, normalmente, através de filtragem colaborativa, abordagem complexa computacionalmente. Para mitigar esse problema da esparsidade de dados, foi proposto o uso de dois conceitos diferentes de *ratings*: *ratings* implícitos e *ratings* explícitos. Os *ratings* implícitos são computados e, por definição, não podem ser desconhecidos. Os *ratings* explícitos são os que o usuário do sistema explicitamente fornece.

Com o conceito de *rating* implícito, é possível computar um critério de uma forma personalizada para a aplicação de domínio, diferente de uma abordagem que prediz *ratings* baseado em filtragem colaborativa. Esse conceito permite, portanto, que aplicações com informações geográficas possam ter um critério de distância entre o usuário e determinado local, por exemplo, ou permita que aplicações de filme tenham um critério baseado em quão recente é determinado filme.

Os experimentos para verificar os efeitos da esparsidade de dados mostram que, de fato, ao se utilizar uma estratégia sem predição de *ratings* explícitos desconhecidos, é possível ter um ajuste mais rápido. Além disso, o uso de *ratings* implícitos permite manter uma precisão tão boa quanto a precisão das recomendações feitas pelo melhor dos *baselines*. É evidenciada a vantagem de se utilizar *ratings* implícitos, todavia, ao se analisar um cenário mais esparso. Quando o cenário está mais esparso, a precisão da recomendação feita pelos *baselines* fica muito abaixo da precisão das recomendações feitas pelo presente trabalho. Há, contudo, algumas limitações que merecem ser destacadas.

O escopo dos experimentos desta dissertação se deu em uma base

de dados de filmes, com usuários de filmes e um número fixo de critérios para os itens. Contudo, a Definição 1 prevê a possibilidade de itens possuírem critérios distintos. Tal possibilidade pode ser utilizada em aplicações de múltiplos domínios (FERNÁNDEZ-TOBÍAS et al., 2012). Nesse tipo de aplicação, os mesmos usuários interagem em diferentes domínios e itens vistos em um domínio podem ser utilizados para recomendação de item de outro domínio. Outro ponto com relação aos múltiplos domínios é que, ao se tratar de múltiplos critérios, pode haver critérios diferentes que sejam similares ou que estejam correlacionados. Por exemplo: gênero musical, gênero de filmes e gêneros de jogos podem ser distintos, mas ainda assim terem correlação.

Outro fator não explorado nesta dissertação foi a possibilidade de determinados critérios possuírem fator negativo para a avaliação de determinado usuário. Algum usuário que goste apenas de filmes clássicos e antigos pode não gostar de lançamentos. No conceito atual, a expectativa é que o critério de data de lançamento mais recente possua um peso baixo. Ainda assim, quanto mais recente for o item, mesmo com um peso pequeno para tal critério, melhor será a recomendação desse item ao usuário. Se o sistema de recomendação conhece esse aspecto do usuário, faria mais sentido que, quanto mais recente fosse o filme, menor seria a recomendação. Portanto, um ajuste de pesos levando em consideração pesos negativos poderia fazer sentido em determinados cenários.

Além disso, atualmente os pesos são ajustados em um processo *offline*, o que significa que, conforme o usuário vai navegando no sistema de recomendação, os ajustes para os critérios de pesos podem não ser realizados em tempo real. Levando-se em consideração o exposto, nota-se a possibilidade de extensão desta dissertação. Portanto, para continuidade do presente trabalho, citam-se, alguns trabalhos futuros.

- Levar em consideração que há critérios de um item que quanto mais alto o *rating* do critério, menor deve ser a chance de o item ser recomendado. Para esse problema, pode-se usar pesos negativos para determinados critérios, de acordo com o usuário. Nota-se, nesse cenário, que as definições precisam ser modificadas.
- Testar aplicação em bases de dados diferentes;
- Ajustar pesos para critérios em um sistema de recomendação com múltiplos domínios;
- Identificar similaridade entre critérios, em um sistema de recomendação com múltiplos domínios;

- Permitir que os ajustes possam ser feito em tempo real.

6.1 PUBLICAÇÃO

O desenvolvimento do presente trabalho teve como resultado a publicação do artigo completo *Weight Adjustment for Multi-criteria Ratings in Items Recommendation* [1] para a conferência *22nd Brazilian Symposium on Multimedia and the Web (WebMedia'16)* [2]. Tal artigo foi apresentado no dia 9 de novembro do ano de 2016, na cidade de Teresina, em Piauí.

[1] `http://dl.acm.org/citation.cfm?doid=2976796.2976846`
[2] `http://www6.ifpi.edu.br/webmedia/?page_id=11&lang=pt`

REFERÊNCIAS

ADOMAVICIUS, G.; MANOUSELIS, N.; KWON, Y. Multi-criteria recommender systems. In: *Recommender systems handbook*. [S.l.]: Springer, 2011. p. 769–803.

CHENG, S. et al. Imrank: influence maximization via finding self-consistent ranking. In: ACM. *SIGIR*. [S.l.], 2014. p. 475–484.

COSTA, M.; COUTO, F.; SILVA, M. Learning temporal-dependent ranking models. In: ACM. *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. [S.l.], 2014. p. 757–766.

FERNÁNDEZ-TOBÍAS, I. et al. Cross-domain recommender systems: A survey of the state of the art. In: *Spanish Conference on Information Retrieval*. [S.l.: s.n.], 2012.

GEDIKLI, F.; JANNACH, D. Improving recommendation accuracy based on item-specific tag preferences. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM, v. 4, n. 1, p. 11, 2013.

GUO, G. Integrating trust and similarity to ameliorate the data sparsity and cold start for recommender systems. In: *RecSys*. New York, NY, USA: ACM, 2013. (RecSys '13), p. 451–454. ISBN 978-1-4503-2409-0. <http://doi.acm.org/10.1145/2507157.2508071>.

HAYKIN, S.; NETWORK, N. A comprehensive foundation. *Neural Networks*, v. 2, n. 2004, 2004.

HORN, J.; NAFPLIOTIS, N.; GOLDBERG, D. E. A niched pareto genetic algorithm for multiobjective optimization. In: IEEE. *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. [S.l.], 1994. p. 82–87.

HORST, R. et al. On optimization over the efficient set in linear multicriteria programming. *Journal of optimization theory and applications*, Springer, v. 134, n. 3, p. 433–443, 2007.

HU, J. et al. Personalized tag recommendation using social influence. *Journal of Computer Science and Technology*, Springer, v. 27, n. 3, p. 527–540, 2012.

HUANG, Z.; CHEN, H.; ZENG, D. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *TOIS*, ACM, New York, NY, USA, v. 22, p. 116–142, jan. 2004. ISSN 1046-8188. <http://doi.acm.org/10.1145/963770.963775>.

HWANG, C.-S. Genetic algorithms for feature weighting in multi-criteria recommender systems. In: CITESEER. *Journal of Convergence Information Technology*. [S.l.], 2010.

JANNACH, D.; KARAKAYA, Z.; GEDIKLI, F. Accuracy improvements for multi-criteria recommender systems. In: ACM. *EC'13*. [S.l.], 2012. p. 674–689.

JOACHIMS, T. Svmlight: Support vector machine. *SVM-Light Support Vector Machine http://svmlight. joachims. org/, University of Dortmund*, v. 19, n. 4, 1999.

JOACHIMS, T. Optimizing search engines using clickthrough data. In: ACM. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2002. p. 133–142.

KHARITONOV, E. et al. Using historical click data to increase interleaving sensitivity. In: ACM. *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. [S.l.], 2013. p. 679–688.

KOREN, Y.; BELL, R.; VOLINSKY, C. Matrix factorization techniques for recommender systems. *Computer*, IEEE, v. 42, n. 8, 2009.

KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*. [S.l.]: MIT press, 1992.

LAKIOTAKI K.; MATSATSINIS, N. T. A. Multicriteria user modeling in recommender systems. *IEEE Intelligent Systems*, v. 26, n. 2, p. 64–76, 2011.

LAMONTAGNE, L.; ABI-ZEID, I. Combining multiple similarity metrics using a multicriteria approach. In: *Advances in Case-Based Reasoning*. [S.l.]: Springer, 2006. p. 415–428.

LEE, C.-J.; CROFT, W. B. Cross-language pseudo-relevance feedback techniques for informal text. In: *Advances in Information Retrieval*. [S.l.]: Springer, 2014. p. 260–272.

LI, Y.; YANG, M.; ZHANG, Z. M. Scientific articles recommendation. In: ACM. *CIKM*. [S.l.], 2013. p. 1147–1156.

LIU, L.; MEHANDJIEV, N.; XU, D.-L. Multi-criteria service recommendation based on user criteria preferences. In: ACM. *ACM*. [S.l.], 2011. p. 77–84.

MA, H. On measuring social friend interest similarities in recommender systems. In: ACM. *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. [S.l.], 2014. p. 465–474.

MANOUSELIS, N.; COSTOPOULOU, C. Analysis and classification of multi-criteria recommender systems. *World Wide Web*, Springer, v. 10, n. 4, p. 415–441, 2007.

MENG, X.; SHAO, J. et al. Semantic approximate keyword query based on keyword and query coupling relationship analysis. In: ACM. *CIKM*. [S.l.], 2014. p. 529–538.

MINKOV, E. et al. Collaborative future event recommendation. In: ACM. *Proceedings of the 19th ACM international conference on Information and knowledge management*. [S.l.], 2010. p. 819–828.

MONTGOMERY, D. C. *Design and analysis of experiments*. [S.l.]: John Wiley & Sons, 2008.

MOSHFEGHI, Y.; PIWOWARSKI, B.; JOSE, J. M. Handling data sparsity in collaborative filtering using emotion and semantic based features. In: *SIGIR*. New York, NY, USA: ACM, 2011. (SIGIR '11), p. 625–634. ISBN 978-1-4503-0757-4. <http://doi.acm.org/10.1145/2009916.2010001>.

NILASHI, M.; IBRAHIM, O. bin; ITHNIN, N. Multi-criteria collaborative filtering with high accuracy using higher order singular value decomposition and neuro-fuzzy system. *Knowledge-Based Systems*, Elsevier, v. 60, p. 82–101, 2014.

PALANIVEL, K.; SIVAKUMAR, R. A study on implicit feedback in multicriteria e-commerce recommender system. *JECR*, JECR, v. 11, n. 2, p. 140–156, 2010.

RENDLE, S. et al. Bpr: Bayesian personalized ranking from implicit feedback. In: AUAI PRESS. *AUAI*. [S.l.], 2009. p. 452–461.

RICCI, F. et al. *Recommender Systems Handbook*. 1st. ed. New York, USA: Springer-Verlag New York, Inc., 2010. ISBN 0387858199, 9780387858197.

ROUSSEAU, F.; VAZIRGIANNIS, M. Graph-of-word and tw-idf: new approach to ad hoc ir. In: ACM. *CIKM*. [S.l.], 2013. p. 59–68.

SARWAR, B. et al. Item-based collaborative filtering recommendation algorithms. In: ACM. *WWW*. [S.l.], 2001. p. 285–295.

SZELAG, M.; GRECO, S.; SŁOWIŃSKI, R. Variable consistency dominance-based rough set approach to preference learning in multicriteria ranking. *Information Sciences*, Elsevier, v. 277, p. 525–552, 2014.

WANG, R.; KWONG, S. Active learning with multi-criteria decision making systems. *Pattern Recognition*, Elsevier, v. 47, n. 9, p. 3106–3119, 2014.

ZHANG, M. et al. Addressing cold start in recommender systems: A semi-supervised co-training algorithm. In: ACM. *SIGIR*. [S.l.], 2014. p. 73–82.

ZHANG, Y. et al. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In: ACM. *SIGIR*. [S.l.], 2014. p. 83–92.

ZHANG, Y. et al. Understanding the sparsity: Augmented matrix factorization with sampled constraints on unobservables. In: *CIKM*. New York, NY, USA: ACM, 2014. (CIKM '14), p. 1189–1198. ISBN 978-1-4503-2598-1. <http://doi.acm.org/10.1145/2661829.2661976>.